UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

MÉMOIRE D'ÉTUDIANT

CASE POSTALE 6128, STATION A, MONTRÉAL, QUÉBEC, H3C 3J7

AMELIORATION DE LA CONVERGENCE DE L'ALGORITHME DE FRANK ET WOLFE APPLIQUE AU PROBLEME D'AFFECTATION DE TRAFIC AVEC DEMANDES FIXES

par

Marc Shaigetz

Université de Montréal

Centre de recherche sur les transports - Publication # 150

Novembre 1978

Dépôt légal - 4e trimestre 1979

SOMMAIRE

Les objectifs du rapport de stage qui suit se résument à améliorer le temps d'exécution et la vitesse de convergence d'un algorithme (développé par Frank et Wolfe [7] basé sur l'approximation linéaire d'une fonction économique convexe, sous des contraintes linéaires.

On s'intéresse aux applications se rapportant au problème de minimisation des coûts pour un réseau à plusieurs produits et plus particulièrement, au problème d'affectation de trafic (à moindre coût pour l'usager), avec demandes fixes.

En considérant l'algorithme de Frank et Wolfe dans le contexte d'une stratégie de restriction et de linéarisation interne du domaine (formé par des contraintes linéaires), on adapte, en s'inspirant de Wolfe [23], une nouvelle direction de descente ("Away step"), en plus d'améliorer les performances de calcul des directions de descente.

TABLE DES MATIÈRES

Pages

TABLE DES TABLEAUX ET FIGURES iii

CHAPITRE PREMIER

Introduction. 1
I.1 Définition et formulation générale du problème. 1
I.2 Méthode des directions réalisables. 1
I.3 Algorithme de Frank et Wolfe. 8

CHAPITRE DEUXIÈME

Extension de l'algorithme de Frank et Wolfe 10
II.1 Linéarisation interne 10
II.2 Stratégie de restriction. 11
II.3 Procédure basée sur la linéarisation interne et la stratégie de restriction. 14
II.4 Procédure de Frank et Wolfe dans le contexte de linéarisation interne et de restriction 15
II.5 Algorithme général basé sur la linéarisation interne et la restriction 19
II.6 Extension de l'algorithme de Frank et Wolfe basée sur plusieurs directions de descente. 21
II.7 Algorithme basé sur plusieurs directions de descente. . . . 27

CHAPITRE TROISIÈME

Modèle de réseau pour plusieurs produits à coût minimum 29
III.1 Définitions et formulation. 29
III.2 Identification au problème général. 31
III.3 Stratégie de restriction. 42
III.4 Adaptation d'un "Away step" restrictif. 47
III.5 Adaptation d'un "Toward step" restrictif. 51

CHAPITRE QUATRIÈME

Algorithme basé sur plusieurs directions de descente pour le modèle de réseau. 54

CHAPITRE CINQUIÈME

Résultats 60
V.1 Essais sur des problèmes pratiques. 60
V.2 Discussion des critères d'arrêt et de partage entre les types d'itérations. 64
V.3 Conclusion. 65

CHAPITRE SIXIÈME

Amélioration du temps de calcul de la direction de descente de Frank et Wolfe 69
VI.1 Introduction. 69
VI.2 Terminologie. 70
VI.3 Représentation du réseau. 71
VI.4 Motivation du choix de l'algorithme 71
VI.5 Algorithme de Dijkstra. 73
VI.6 Amélioration de l'algorithme de Ford. 76
VI.7 Comparaison des performances. 84

CHAPITRE SEPTIÈME

Conclusion. 87

ANNEXES

A - Exemple de formulation. 90

B - Programme utilisé pour le réseau de la ville de Winnipeg. . . 95

C - Programme utilisé pour le réseau de la ville de Hull. 113

D - Description des variables 136

REMERCIEMENTS 140

BIBLIOGRAPHIE 141

TABLE DES TABLEAUX ET FIGURES

Tableau 1 - Comparaison de solutions optimales obtenues à l'aide de trois algorithmes différents 61

Tableau 2 - Comparaison de deux algorithmes de calcul des chemins les plus courts 86

Figure 1 80

CHAPITRE PREMIER

INTRODUCTION

I.1 Définition et formulation générale du problème

Soit $f(x)$: $E^n \to E^1$ une fonction convexe sur $\underline{\overline{X}} \subset E^n$ tel que $\underline{\overline{X}}$ forme un ensemble convexe de contraintes linéaires. On s'intéresse au problème général:

$$(1) \qquad \text{Minimiser } f(x)$$
$$x \geq 0, \quad x \in \underline{\overline{X}}$$

On peut exprimer $\underline{\overline{X}}$ par la notation $A\,x = b$ où A, b sont des matrices m x n et m x 1 respectivement. Cette formulation facilite la compréhension du développement des algorithmes qui suivent; on introduira au troisième chapitre le modèle de réseau tout en établissant la correspondance avec le modèle général.

I.2 Méthode des directions réalisables

Les méthodes des directions réalisables se classifient parmi les méthodes primales car elles recherchent de façon itérative la solution du problème en se déplaçant dans le domaine réalisable, d'un point à l'autre. A chaque étape, la solution demeure réalisable, et la valeur de la fonction économique décroît. Ces méthodes s'appliquent à la plupart des problèmes nonlinéaires et possèdent plusieurs avantages.

En effet, si l'algorithme s'arrête à cause d'une limite de temps, la solution obtenue s'avère souvent acceptable pour un problème pratique. De plus, le processus itératif converge vers au moins un minimum local. En-

fin, ces méthodes ne reposent pas sur des propriétés particulières (telle que la convexité) et sont efficaces pour des contraintes linéaires sur les variables.

Il existe de nombreuses méthodes pour résoudre des problèmes de programmation nonlinéaire pouvant être classifiées parmi les méthodes de "directions réalisables". Elles diffèrent entre elles par la façon de choisir le point initial, la direction et la longueur du déplacement dans cette direction. Parmi ces méthodes, on retrouve l'approximation linéaire de Frank et Wolfe, la méthode du gradient projetté de Rosen, la méthode du gradient réduit de Wolfe et le simplexe convexe.

L'exposé qui suit s'inspire de Ferland [5], Frank et Wolfe [7]. Luenberger [16] et Zoutendijk [25]. On se limite à des contraintes d'égalité car dans le modèle de réseau les contraintes d'équilibre s'écrivent sous cette forme.

Deux lemmes sont nécessaires au développement de la méthode des directions réalisables:

I.2.1 Lemme:

Soient $\overline{X}$, $f(x) \in C^1$ sur $\overline{X}$

$x \in \overline{X}$, $d \in E^n$, alors on a:

$$\lim_{t \to 0} \frac{f(x+td) - f(x)}{t} = (\nabla f(x), d)$$

Preuve: Le développement de Taylor de $f(x+td)$ donne:

$$f(x) + (\nabla f(x), d) + \alpha(x,td) \| td \| .$$

Si $t \neq 0$ alors on a la relation:

$$\frac{f(x+td) - f(x)}{t} \pm [\alpha(x,td)\|d\|] = (\nabla f(x),d)$$

D'où

$$\lim_{t\to 0} \frac{f(x+td) - f(x)}{t} = (\nabla f(x),d) .$$

Observons que l'expression $(\nabla f(x),d)$ indique le taux de changement de la valeur de $f(x)$ dans le voisinage de x , quand on s'éloigne de x dans la direction d, $\|d\|_2 = 1$. Le lemme suivant permet de reconnaître une direction d de descente.

I.2.2 Lemme:

Soient $\overline{X}$, $f(x) \in C^1$ sur $\overline{X}$, $x \in \underline{X}$, $d \in E^n$;

de plus si on a $(\nabla f(x),d) < 0$ alors il existe $\xi > 0, \xi \in E^1$,

tel que pour tout $t \in E^1$, $0 < t \leq \xi$

$$f(x+td) < f(x)$$

preuve: Par hypothèse, on a:

$$(\nabla f(x),d) = \lim_{t\to 0} \frac{f(x+td) - f(x)}{t}$$

Par la notion de limite, il existe $\xi > 0$ tel que pour tout $t \neq 0$, $-\xi \leq t \leq \xi$

$$\frac{f(x+td) - f(x)}{t} < 0$$

En choisissant $0 < t \leq \xi$,

$$f(x+td) < f(x) .$$

I.2.3 Objectif

Etant donné un point $\bar{x}$ réalisable, $\nabla f(\bar{x}) \neq 0$ et une fonction économique qui décroît un certain temps quand on s'éloigne de $\bar{x}$ dans la direction $d = -\nabla f(\bar{x})$, alors on a $(\nabla f(\bar{x}), -\nabla f(\bar{x}) < 0$, et en utilisant le lemme I.2.2, il existe $\xi > 0$ tel que pout tout t, $0 < t < \xi$, on a:

$$f(\bar{x}) - t\nabla f(\bar{x}) < f(\bar{x}) .$$

La procédure qui suit génère, à partir d'une solution réalisable initiale, une suite de solutions réalisables $\{x^{\ell}\}$, telles que $f(x^{\ell+1}) < f(x^{\ell})$

I.2.4 Problème: On cherche à:

(2) Minimiser $f(x)$

sujet à

$$(A^i, x) = b_i \qquad i \in I = \{1,\ldots,m\}$$

$$x_j \geq 0 \qquad j \in J = \{1,\ldots,n\}$$

où,

A^i désigne la $i^{ème}$ contrainte et x_j la $j^{ème}$ variable.

I.2.5 Hypothèse A:

Soient $\underline{X}$ le domaine réalisable, $x^{\ell} \in \underline{X}$ et $d^{\ell} = -\nabla f(x^{\ell}) \neq 0$; faisons l'hypothèse qu'il existe $\xi > 0$ tel que

$$x^{\ell+1} = x^{\ell} + t^{\ell} d^{\ell} \in \underline{X}$$

pour tout t^{ℓ}, $0 \leq t^{\ell} \leq \xi$, (donc $x^{\ell+1}$ est dans le domaine réalisable) deux situations peuvent se produire:

A-1: On a $\xi < \infty$; alors on choisit $x^{\ell+1}$ tel que:

$$f(x^{\ell+1}) = \underset{\theta}{\text{Min}} \{f(x) : x = \theta x^{\ell} + (1-\theta)(x^{\ell} + t^{\ell} d^{\ell}),\ 0 \le \theta \le 1\}.$$

A-2 On a $\xi = \infty$; alors on choisit $x^{\ell+1}$ tel que

$$f(x^{\ell+1}) = \underset{\theta}{\text{Min}} \{f(x) : x = x^{\ell} + \theta\, d^{\ell},\ \theta > 0\}.$$

En somme, étant donné une solution initiale x^{ℓ}, on se déplace le long de la projection du gradient de la fonction économique, évalué en x^{ℓ}. On choisit $x^{\ell+1}$ tel que $f(x^{\ell+1})$ soit minimale sachant bien que $x^{\ell+1}$ peut se trouver à l'intérieur ou à la frontière du domaine. D'autre part, si $x^{\ell+1}$ n'existe pas, alors $f(x)$ n'est pas bornée sur son domaine.

On reprend ce processus pour générer $x^{\ell+1}$, $x^{\ell+2}$,..., jusqu'à ce qu'un critère d'arrêt soit satisfait. Nous allons maintenant chercher à déterminer ξ.

Valeur de ξ:

La valeur de ξ identifie la borne supérieure sur t^{ℓ}, telle que les contraintes soient satisfaites pour $x^{\ell+1}$.

Définissons $J^{\ell} \subset J$ tel que $j \in J^{\ell}$ si $x_j^{\ell} \neq 0$. On veut se déplacer dans la direction d^{ℓ} et rester dans le domaine $\overline{X}$; on doit avoir

$$(*) \quad (A^i, x^{\ell} + t^{\ell} d^{\ell}) = b_i \qquad i \in I$$

$$(**) \quad x_j^{\ell} + t^{\ell} d_j^{\ell} \ge 0 \qquad j \in J^{\ell}$$

Pour déterminer ξ, on calcule:

$$\xi = \begin{cases} \underset{j}{\text{Min}} \{-x_j^\ell / d_j^\ell\}, \; d_j^\ell < 0, \; j \in J^\ell \\ \infty \text{ si } d_j^\ell \geq 0, \; j \in J \end{cases}$$

Avec l'hypothèse A et $0 < t \leq \xi$, les contraintes (*) et (**) sont satisfaites.

I.2.6 Hypothèse B:

Soient $\overline{X}$ le domaine réalisable, $x^\ell \in \overline{X}$, et $d^\ell = -\nabla f(x^\ell) \neq 0$; faisons maintenant l'hypothèse que l'on sort du domaine réalisable dès qu'on se déplace dans la direction d^ℓ, c'est-à-dire que pout tout $\xi > 0$, on a

$$x^{\ell+1} = x^\ell + \xi \, d^\ell \notin \overline{X} .$$

Encore deux situations peuvent se produire:

B-1: Si pour toute direction d, on a la relation $(\nabla f(x^\ell), d) > 0$ alors la fonction économique atteint son minimum local au point x^ℓ.

B-2: D'autre part, s'il existe plusieurs directions réalisables d^i ($\|d^i\|_2 = 1$), on retient celle qui possède le plus grand taux de diminution:

$$d^* = \underset{i}{\text{Min}} \{(\nabla f(x), d^i)\}$$

où $|d^i| = 1$.

Comme on verra plus loin, nous aurons le choix dans une version améliorée de l'algorithme de Frank et Wolfe, entre trois directions de descente (Toward, Toward restrictif, Away restrictif) et nous dérogerons à la règle du plus grand taux de diminution à cause des coûts relatifs de calcul de ces directions.

On sait qu'une direction d^ℓ est réalisable si et seulement si, pour

tout t^ℓ, $0 < t^\ell \leq \xi$

1 - $(A^i, x^\ell + t^\ell d^\ell) = b_i \Rightarrow (A^i, d) = 0 \qquad i \in I$

2 - $x_j^\ell + t^\ell d_j^\ell \geq 0 \Rightarrow d_j^\ell \geq 0 \qquad j \notin J^\ell$

3 - $x_j^\ell + t^\ell d_j^\ell \geq 0 \qquad j \in J^\ell$

Donc, pour déterminer la direction possédant le plus grand taux de diminution, il faut résoudre le sous-problème:

$$(3) \quad \underset{\{d^\ell\}}{\text{Minimiser}} \ (\nabla f(x^\ell), d^\ell)$$

sujet à:

$$(A^i, d^\ell) = 0 \qquad i \in I$$

$$d_j^\ell \geq 0 \qquad j \notin J^\ell$$

$$\|d\|_2 = 1$$

Soit d* la solution optimale de (3). On se trouve à un minimum de f(x) au point x^ℓ si on a la relation $(\nabla f(x^\ell), d^*) \geq 0$.

Si par contre, $(\nabla f(x^\ell), d^*) < 0$, alors la plus grande valeur de t^ℓ pour laquelle $x^\ell + t^\ell d^*$ reste dans le domaine réalisable, s'obtient comme précédemment en calculant ξ (cf. hypothèse A, valeur de ξ).

On poursuit pour $x^{\ell+1}$, $x^{\ell+2}$, ... jusqu'à ce qu'un critère d'arrêt soit satisfait.

Pour résoudre le sous-problème (3) comportant une contrainte non linéaire ($\|d\|_2 = 1$), il existe plusieurs techniques, voir Zangwill

[24], Zoutendijk [25]. Cependant, pour l'algorithme développé, on se limitera à des directions dans le sens opposé du gradient ou à des directions de descentes composées à partir d'un ensemble de solutions antérieures, évitant ainsi la non linéarité. En particulier, résoudre le sous-problème (3) reviendra à un calcul des chemins les plus courts.

I.3 Algorithme de Frank et Wolfe:

Nous donnons ici une version de l'algorithme pour le problème exprimé sous sa forme générale:

$$(4) \qquad \text{Minimiser } f(x) \quad x \geq 0,\ x \in \underline{X}$$

où,

- x est un vecteur $n \times 1$
- $f(x): x \to E^1$ est convexe et continûment différentiable sur le domaine convexe $\underline{X}$.

Etape 0: trouver une solution réalisable $x^0 \in \underline{X}$; $\ell = 0$;

Etape 1: résoudre le sous-problème: trouver $\bar{x}^\ell$ tel que

$$\nabla f(x^\ell)\ (x - x^\ell)$$

soit minimum pour $x = \bar{x}^\ell$ et $\bar{x}^\ell \in \underline{X}$; si $\nabla f(x^\ell)\ (\bar{x}^\ell - x^\ell) \geq 0$ alors on s'arrête, sinon on passe à l'étape 2.

Etape 2: résoudre pour t^ℓ tel que

$$f(x^\ell + t\ (\bar{x}^\ell - x^\ell))$$

soit minimum lorsque $t = t^\ell$. La nouvelle valeur de x^ℓ s'obtient:

$$x^{\ell+1} = x^\ell + t^\ell\ (\bar{x}^\ell - x^\ell);$$

On met à jour le compteur d'itération: $\ell = \ell + 1$ et on retourne à l'étape 1.

En fait, l'algorithme de Frank et Wolfe est une méthode de directions réalisables basée sur l'approximation linéaire de la fonction économique. L'expansion de Taylor de f(x) autour du point $x^{\ell+1}$ peut s'écrire:

$$f(x^{\ell+1}) = f(x^{\ell} + t^{\ell}(x - x^{\ell}))$$

$$= f(x^{\ell}) + t^{\ell}(\nabla f(x^{\ell}), x - x^{\ell}) + \alpha(x, t^{\ell}(x-x^{\ell}))\| t^{\ell}(x - x^{\ell})\|$$

$$\simeq f(x^{\ell}) + t^{\ell}(\nabla f(x^{\ell}), x - x^{\ell}))$$

Et

$$\underset{x \in \underline{X}}{\text{Min}}\ f(x^{\ell+1})$$

revient à résoudre le sous problème:

$$\text{(I.3.1)} \qquad \underset{x \in \overline{X}}{\text{Min}}\ (\nabla f(x^{\ell}), x) + \text{constante}$$

Dans la section suivante, nous allons considérer la méthode de Frank et Wolfe dans le contexte d'une stratégie de restriction et de linéarisation interne du domaine $\underline{X}$.

CHAPITRE DEUXIEME

Extension de l'algorithme de Frank et Wolfe

Les lignes qui suivent s'inspirent de l'article de C.A. Holloway [12] dans lequel il montre que l'on peut déduire la méthode de Frank et Wolfe du cas général, consistant en une stratégie de restriction et de linéarisation interne du domaine formé par des contraintes. Wolfe [23] introduit un algorithme basé sur le choix, à chaque itération, entre deux directions de descente, selon le critère du plus grand taux de diminution de la fonction économique. Nous retiendrons et développerons, dans ce chapitre, les informations nécessaires pour résoudre le problème (1).

II.1 Linéarisation interne:

La linéarisation interne s'applique aussi bien à une fonction convexe (ou concave), qu'à un domaine convexe $\overline{X}$ (voir Geoffrion [8]). L'approximation de $\overline{X}$ peut se faire à partir d'un ensemble de L points notés x^j, $x^j \in \overline{X}$, $j = 0, \ldots, L-1$ que l'on appelle points de base de la linéarisation interne de $\overline{X}$. En effet, l'ensemble

$$\{y : y = \sum_{j=0}^{L-1} \alpha_j x^j \ , \ \sum_{j=0}^{L-1} \alpha_j = 1, \ \alpha_j \geq 0\}$$

est convexe et constitue l'approximation de $\overline{X}$.

Pour le modèle de réseau qui sera abordé dans le chapitre suivant, le domaine est un polytope (sous-ensemble de E^n formé de l'intersection d'un nombre fini de demi-espaces fermés dans E^n). Un polytope borné s'appelle polyèdre. Lorsque les points de base $\{x^j\}$ se confondent avec

tous les points extrêmes du polyèdre, la linéarisation interne n'induit par d'erreur. Le théorème qui affirme l'exactitude de la linéarisation interne pour les polyèdres se retrouvent à la référence: Goldman [10]. En pratique, il suffit de se rappeler qu'un ensemble convexe (formé de contraintes linéaires) peut se représenter avec la précision voulue, suivant le nombre de points de base $\{x^j\}$.

On peut donc reformuler le problème initial (1):

$$(5) \quad \underset{\{\alpha\}}{\text{Min}} \; f(\alpha X)$$

sujet à:

$$\sum_{j=0}^{L-1} \alpha_j = 1$$

$$\alpha^j \geq 0 \qquad j = 0, \ldots, L-1$$

où,

$$X = (x^0, x^1, \ldots, x^{L-1})^T$$

$$\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{L-1})$$

II.2 Stratégie de restriction:

La stratégie de restriction s'utilise pour la résolution de problèmes comportant un grand nombre de variables non négatives. L'idée de base se résume ainsi: à chaque itération on définit un ensemble de variables qui seront fixées à 0; si la solution résultante ne satisfait pas les conditions d'optimalité du problème en question, il s'agit de modifier l'ensemble des variables fixées à 0. La stratégie de restriction évite de spécifier tous les points de base $\{x^j\}$. Voir Geoffrion [8] pour plus d'information.

Soient P l'ensemble (non vide) des indices pour retrouver les points de base de la linéarisation interne du domaine convexe $\underline{X}$, et $\Pi \in E^1$, $\lambda \in E^n$, des multiplicateurs; on peut alors récrire le système (5):

(6) Min $f(\alpha X)$

sujet à

$$\sum_{j=0}^{L-1} \alpha = 1 \qquad \text{multiplicateur } \Pi$$

$$\alpha_j \geq 0 \quad j \in P$$
$$\alpha_j = 0 \quad j \notin P \qquad \text{multiplicateur } \lambda$$

La stratégie de restriction se déroule comme suit:

étant donné un ensemble P, on vérifie si la solution optimale du problème (6), $\hat{\alpha}$,répond aussi aux conditions d'optimalité du problème (5); si oui, alors la procédure se termine, sinon on forme un sous-ensemble de variables incluant au moins une variable (la $j^{\text{ième}}$) dont le multiplicateur λ^j est négatif; les variables de ce sous-ensemble ne sont plus restreintes à une valeur nulle; d'autre part, si la valeur de la fonction économique décroît, alors on ajoute l'ensemble des variables $\{\alpha_j : \alpha_j = 0, j \in P\}$ à l'ensemble des variables restreintes. On reprend la procédure.

Le lemme qui suit établit les conditions d'optimalité d'une solution $\hat{\alpha}$, $\hat{\Pi}$, $\hat{\lambda}$ du problème (6).

II.2.1 Lemme:

Si f(x) est convexe et continûment différentiable sur $\underline{X}$, et $\hat{\alpha}$ est la solution du problème (6), alors les multiplicateurs optimaux $\hat{\Pi}$, $\hat{\lambda}$ existent

de plus, $\hat{\alpha}$ est aussi la solution optimale du problème (5) (et $\hat{x} = \hat{\alpha}X$ pour le problème original (4)) si:

$$(II.2.2) \qquad \hat{\lambda} = \min_{x \in \underline{X}} \; [\hat{\Pi} + \nabla f(\hat{x})x]$$

est non négatif.

Preuve: Les conditions de Kuhn-Tucker pour le problème (6) s'énoncent (Luenberger [16]):

$$\nabla f(\hat{\alpha}X) + \hat{\Pi}\, \nabla h(\hat{\alpha}) + \hat{\lambda}\, \nabla g(\hat{\alpha}) = \tilde{0}$$

$$\hat{\lambda} g(\hat{\alpha}) = \tilde{0}$$

où,

$$h(\hat{\alpha}) = \sum_{j=0}^{L-1} \hat{\alpha}_j - 1$$

$$g(\hat{\alpha}) = -\hat{\alpha} \leq \tilde{0}$$

$$\nabla f(\hat{\alpha}X) = [\nabla f(\hat{x})\, x^1, \ldots, \nabla f(\hat{x})\, x^{L-1}]$$

$$\hat{\Pi}\, \nabla h(\hat{\alpha}) = [\hat{\Pi}, \ldots, \hat{\Pi}]$$

$$\nabla g(\hat{\alpha}) = [-1, \ldots, -1]$$

$$\tilde{0} = [0, \ldots, 0]$$

Les multiplicateurs existent et pour chaque composante on a:

$$\hat{\lambda}^j = \hat{\Pi} + \nabla f(\hat{x})\, x^j.$$

Les conditions de Kuhn-Tucker sont suffisantes et identiques pour les problèmes (5) et (6) sauf en ce qui a trait aux multiplicateurs $\hat{\lambda}^j$, lesquels sont non négatifs, pour tout j du problème (5).

Donc, il faut que le vecteur $\hat{\lambda}$ soit non négatif pour chaque composante j, ce qui revient à dire que

$$\hat{\lambda}^j = \hat{\Pi} + \nabla f(\hat{x})x^j$$

doit être non négatif et ce, pour tout j. Par conséquent, $\hat{\alpha}$, $\hat{\Pi}$, $\hat{\lambda}$ satisfont les conditions d'optimalité de Kuhn-Tucker pour le problème (5).

Le lemme II.2.1 donne la condition suffisante pour que la solution courante du problème (5) soit optimale et détermine le critère d'arrêt. Si la solution du sous-problème (II.2.2) est négative pour une composante j, alors on doit enlever au moins une variable correspondant à ce $j^{ième}$ multiplicateur négatif qu'on avait restreint auparavant.

II.3 Procédure basée sur la linéarisation interne et la stratégie de restriction

Une procédure basée sur la linéarisation interne du domaine et sur la stratégie de restriction peut prendre la forme:

Etape 0: on débute avec un ensemble de points de base:

$$\{x^j : j \in P\}$$

Etape 1: résoudre le problème (6) et trouver ainsi le vecteur optimal $\hat{\alpha}$.

Etape 2: résoudre le sous-problème (II.2.2) et obtenir $\bar{x}$; si $\bar{\lambda} \geq \tilde{0}$ alors on s'arrête
sinon on ajoute $\bar{x}$ à l'ensemble

$$\{x^j : j \in P\}$$

et retourne à l'étape 1.

Pour l'algorithme qui sera développé plus loin, l'étape initiale sera une génération des premiers points extrêmes x^j, l'étape 1 consistera en une recherche d'un déplacement optimal dans une direction de descente et l'étape 2 se résumera à former une nouvelle direction de descente à l'aide de $\bar{x}$, obtenu soit par la résolution d'un sous-problème définit par une stratégie de restriction et de linéarisation, soit par la résolution d'un sous-problème définit par une approximation linéaire de la fonction économique (voir section I.3).

Enfin, remarquons que cet algorithme est un cas particulier où l'on choisit de limiter l'ensemble P à deux éléments à chaque itération.

II.4 Procédure de Frank et Wolfe dans le contexte de linéarisation interne et de restriction

A l'aide des deux lemmes suivants, on pourra déduire la procédure de Frank et Wolfe dans le cas général de linéarisation interne du domaine et d'une stratégie de restriction.

II.4.1 Lemme:

Si $f(x)$ est continûment différentiable sur le domaine $\underline{\bar{X}}$ et $\hat{\alpha}$ est la solution optimale du problème (6), alors les $\hat{\lambda}^j$ associés aux $x^j \in \underline{\bar{X}}$ sont uniques et donnés par l'expression:

$$\lambda^j = \nabla f(\hat{x})\ (x^j - \hat{x})$$

Preuve:

De la preuve du lemme II.2.1, on a la relation:

(A) $\quad \hat{\lambda}^j = \hat{\Pi} + \nabla f(\hat{x})\, x^j.$

Les conditions de Kuhn-Tucker se résument à:

i) $\hat{\lambda} \geq 0$ pour tout $j \in P$

ii) $\hat{\Pi} + \nabla f(\hat{x}) x^j - \hat{\lambda}^j = 0$ pour tout $j \in P$

iii) $\sum_{j \in P} \hat{\lambda}^j \hat{\alpha}_j = 0.$

On peut éliminer λ en utilisant la contrainte (ii). La contrainte (iii) devient:

. $\sum_{j \in P} [\hat{\Pi} + \nabla f(\hat{x})\, x^j]\, \hat{\alpha}_j = 0$

. $(\hat{\Pi}) \sum_{j \in P} \hat{\alpha}_j + \nabla f(\hat{x})\, \hat{x} = 0$

. $\hat{\Pi} = -\nabla f(\hat{x})\, \hat{x}$

En substituant la valeur de $\hat{\Pi}$ dans la relation (A) on obtient:

$$\lambda^j = -\nabla f(\hat{x})\, \hat{x} + \nabla f(\hat{x})\, x^j$$

$$= \nabla f(\hat{x})\, (x^j - \hat{x})$$

Le multiplicateur $\hat{\Pi}$ étant unique, $\hat{\lambda}^j$ est uniquement déterminé.

Ainsi, le sous-problème:

(II.4.2) $\quad \underset{x \geq 0,\ x \in \underline{X}}{\text{Minimiser}} \quad \nabla f(\hat{x})\, (x - \hat{x})$

donne la direction utilisée par l'algorithme de Frank et Wolfe tel que décrit dans la section I.3, appliqué à notre problème de minimisation d'une fonction convexe, assujetti à des contraintes linéaires. Cette direction de descente est $d = (\bar{x} - \hat{x})$, tel que $\bar{x}$ résoud le sous-problème (II.4.2).

II.4.3 Lemme:

Soit ℓ un indice désignant le numéro d'itération. Si la solution du problème (6) à l'itération ℓ, $\tilde{\alpha}^{\ell}$, n'est pas optimale pour le problème (5) à l'itération ℓ, alors

$$f(\hat{x}^{\ell+1}) < f(\hat{x}^{\ell})$$

si et seulement si

$\bar{x}^{\ell}$ associé avec un multiplicateur négatif est inclu dans l'ensemble $\{x^j : j \in P^{\ell+1}\}$ pour le problème (6) à l'itération $\ell+1$.

Preuve:

Si aucun point de base associé à un multiplicateur négatif n'est inclu dans l'ensemble $\{x^j : j \in P^{\ell+1}\}$, alors $\tilde{\alpha}^{\ell}$ et les multiplicateurs du problème (6) à l'itération ℓ satisfont les conditions d'optimalité de Kuhn-Tucker, par conséquent

$$f(\hat{x}^{\ell+1}) = f(\hat{x}^{\ell})$$

Pour la condition suffisante, il s'agit de remarquer que l'ensemble $P^{\ell+1}$ inclut l'ensemble P^{ℓ} (i.e. $\{j : j \in P^{\ell}, \alpha_j > 0\}$), donc $\tilde{\alpha}^{\ell}$ demeure réalisable pour le problème (6) à l'itération $\ell+1$ et

$$f(\hat{x}^{\ell+1}) \leq f(\hat{x}^{\ell})$$

Considérons le cas où $f(\bar{x}^{\ell+1}) = f(\bar{x}^{\ell})$. la solution $\bar{\alpha}^{\ell}$ est aussi optimale dans le problème (3) à l'itération $\ell+1$. Pour ce problème, il existe des multiplicateurs associés aux x^j, $j \in p^{\ell+1}$ qui sont non négatifs. Mais pour le lemme II.4.1, ces multiplicateurs uniques s'obtiennent par l'expression:

$$\lambda^j = \nabla f(\hat{x}) \ (x^j - \hat{x})$$

Donc, $\nabla f(\hat{x}) \ (\bar{x}^{\ell} - \hat{x}) \geq 0$, et on a une contradiction avec l'hypothèse voulant que $\bar{x}^{\ell}$ correspond à un multiplicateur négatif; ainsi, $f(\bar{x}^{\ell+1}) < f(\bar{x}^{\ell})$.

Il est maintenant possible d'interpréter la procédure de Frank et Wolfe dans un contexte plus général: seul le domaine de la fonction économique subit une linéarisation interne; on choisit à chaque itération le multiplicateur le plus négatif et l'ensemble P est défini de façon à restreindre toutes les variables sauf $\bar{x}$ et $\hat{x}$.

En effet, soit $\bar{x}^{\ell}$ la solution du sous-problème (II.4.2). Par le lemme précédent, en enlevant $\bar{x}^{\ell}$ de l'ensemble restrictif on s'assure d'une diminution de la valeur de la fonction économique.

Mais, puisque la valeur de la fonction économique décroît, on peut transférer dans l'ensemble restrictif toutes les variables du problème (6) satisfaisant la non négativité avec l'égalité, c'est-à-dire,
$\bar{\alpha}_j^{\ell} = 0 \Rightarrow j \notin p^{\ell+1}$.

Considérons $\bar{\alpha}^{\ell}$ une solution du problème (6) à l'itération ℓ. On a $\bar{x}^{\ell} = \bar{\alpha}^{\ell} X$ Il y a deux possibilités: $\bar{x}^{\ell}$ est membre du domaine déter-

miné par les points de base $\{x^j : j \in P^\ell\}$ ou non.

Dans le second cas, on doit inclure $\hat{x}$ dans l'ensemble $P^{\ell+1}$, ce qui implique une recherche d'un $k \in P^\ell$ tel que l'ensemble

$$\{\tilde{\alpha}_j^\ell = 0,\ j \neq k \quad \text{et} \quad \tilde{\alpha}_k^\ell = 1\}$$

soit une solution du problème (6) à l'itération ℓ.

Donc, on peut ajouter tous les $\tilde{\alpha}_j^\ell$, $j \neq k$ à l'ensemble restrictif laissant uniquement les points de base associés aux points extrêmes $\hat{x}^\ell$ et $\bar{x}^\ell$ sans restriction dans le problème (6) à l'itération $\ell+1$ ce qui revient à une minimisation le long du segment de droite $(\bar{x}^\ell - \hat{x}^\ell)$, exactement comme dans la méthode de Frank et Wolfe.

Il est maintenant possible de reviser l'algorithme de Frank et Wolfe.

II.5 Algorithme général basé sur la linéarisation interne et la restriction

On peut améliorer la performance de convergence de l'algorithme de Frank et Wolfe en inférant sur le choix des variables x^j, $j \in P$ qui seront ajoutées à l'ensemble restrictif, lors de l'itération suivante.

En effet, au lieu de restreindre toutes les variables sauf celle que l'on retrouve sur le segment de droite compris entre $\hat{x}$ et $\bar{x}$, on peut garder tous les x^j, $j \in P$ (ou un sous-ensemble) sans restriction. L'avantage réside dans le fait que l'optimisation porte sur plusieurs directions au lieu d'une seule; cependant le problème (6) devient plus compliqué à résoudre, comme on verra dans la prochaine section.

Le nouvel algorithme se décrit comme suit:

(II.5.1)

Etape 0: Trouver une solution réalisable $\hat{x} \in \underline{X}$; assigner un α_j à $\hat{x}$ et lui donner la valeur 1; tous les autres α_j ont une valeur nulle.

Etape 1: Trouver $\bar{x}$, tel que $\bar{x}$ est solution du sous-problème

$$\text{Minimiser}_{x \geq 0,\ x \in \underline{X}} \nabla f(\hat{x})\ (x - \hat{x})$$

Si la valeur de la fonction économique de ce sous-problème est positive ou nulle alors on s'arrête; sinon, on ajoute $\bar{x}$ à l'ensemble $\{x^j : j \in P\}$ et on continue à l'étape 2.

Etape 2: Résoudre le problème (6) et trouver $\hat{\alpha}$. On a $\hat{x} = \hat{\alpha} X$. Retourner à l'étape 1.

Les théorèmes qui suivent peuvent se démontrer facilement en s'inspirant de l'article de Holloway [12]. Le théorème II.5.2 établit les conditions nécessaires et suffisantes pour vérifier l'optimalité d'une solution en main.

II.5.2 Théorème:

Si $f(x)$ est convexe et continûment différentiable sur le domaine $\underline{X}$ et $\hat{\alpha}$ est la solution du problème (6), alors $\hat{\alpha}$ est aussi optimal pour le problème (5) et de plus $\hat{x} = \hat{\alpha} X$ est optimal pour le problème (1) si et seulement si la valeur optimale du sous-problème (II.4.2) est non négative.

II.5.3 Lemme:

Si f(x) est continûment différentiable sur l'ensemble convexe $\underline{X}$ et on a une séquence $\{x^j\}, j=1,\ldots,\infty$ générée par l'algorithme (II.5.1), alors il ne peut y avoir de sous-séquence $\{x^k\}$, $k \in \Pi$ possédant en même temps les trois propriétés:

i) $x^k \to x^\infty$, $k \in \Pi$

ii) $\bar{x}^k \to \bar{x}^\infty$, $k \in \Pi$

iii) $\nabla f(x^\infty)\ (\bar{x}^\infty - x^\infty) > 0.$

Le théorème suivant établit la convergence de l'algorithme (II.5.1); la notion de stabilité se retrouve dans l'article de Geoffrion [9].

II.5.5 Théorème:

Soit x* une solution du problème (4) telle que ce problème est stable autour de x*. Si f(x) est convexe et continûment différentiable sur l'ensemble compact $\underline{X}$, alors l'algorithme (II.5.1) converge vers une solution optimale du problème (4).

II.6 Extension de l'algorithme de Frank et Wolfe basée sur plusieurs directions de descente

Wolfe dans son article "Convergence theory in nonlinear programming" étudie la convergence de l'algorithm de Frank et Wolfe et propose un nouvel algorithme [23]. Il introduit deux concepts se rapportant à une fonction f(x) sur un domaine convexe $\underline{X}$ dont les dérivées du second ordre sont continues sur $\underline{X}$:

. fonction fortement convexe: s'il existe une constante positive 1/2 a,

constituant une borne inférieure uniforme sur toutes les valeurs propres du Hessien $[\delta^2 f(x)/\delta x_i \delta x_j]$.

. <u>fonction convexe bornée</u>: s'il existe une borne supérieure uniforme 1/2 A sur toutes les valeurs propres du Hessien.

Le taux de convergence s'exprime en fonction de ces constantes. Quand f(x) est fortement convexe et bornée sur le domaine $\overline{X}$ (convexe et borné), Wolfe montre que l'algorithme de Frank et Wolfe possède une borne arithmétique sur le taux de convergence le plus lent ainsi que sur le taux le plus rapide, et ce, lorsque la solution optimale se situe sur la frontière du domaine. Si la solution optimale se trouve à l'intérieur du domaine alors le taux de convergence devient géométrique.

Wolfe montre qu'il s'agit d'avoir suffisamment de directions de descente autour du point $\bar{x}$ pour s'assurer un taux de convergence géométrique. Par conséquent, il est possible de montrer que l'algorithme (II.5.1) possède une telle propriété.

On reproduit ici la modification apportée par Wolfe à l'algorithme original. Il prouve un théorème qui affirme:

si f(x) est fortement convexe et bornée, et $\overline{X}$ est un polyèdre borné, alors l'algorithme qui suit possède une convergence géométrique vers la solution optimale $\bar{x}$.

(II.6.1) Algorithme de Wolfe:

Etape 0: Trouver une solution réalisable $\hat{x} \in \underline{X}$.

Etape 1: Calculer:

. $\mu_1 = \underset{x \in \underline{X}}{\text{Min}} \{\nabla f(\hat{x})\ (x-\hat{x})\} = \nabla f(\hat{x})\ (\bar{x}-\hat{x})$

. $\mu_2 = \underset{x \in \underline{X}}{\text{Max}} \{\nabla f(\hat{x})\ (x-\hat{x})\} = \nabla f(\hat{x})\ (\bar{\bar{x}}-\hat{x})$

Etape 2: Si

$(|\mu_2| > |\mu_1|)$ et $(\lambda^* = \text{Max}\ \{\lambda : \lambda\ \hat{x} + (1-\lambda)\ \bar{\bar{x}} \in \underline{X}\} > 1)$

alors

$y = \lambda^*\ \hat{x} + (1-\lambda^*)\ \bar{\bar{x}}$

(*Away step*)

sinon

$y = \bar{x}$

(*Toward step*)

Etape 3: Déterminer x^* tel que

. $f(x^*) = \underset{t}{\text{Min}}\ \{f(\hat{x} + t(y-\hat{x})) : 0 \leq t \leq 1\}$

. $\hat{x} = x^*$

. retourner à l'étape 1

Wolfe [23] nomme "Toward step" l'étape usuelle consiste à se déplacer d'un point à l'autre, à travers le domaine et subissant toujours l'influence des solutions obtenues antérieurement; l'étape "Away step" désigne une descente vers la frontière du domaine dans une direction de descente qui est l'opposé de la meilleure direction de montée.

Dans l'article "An Improved Linear Approximation for the Network Equilibrium (packet switching) Problem" Florian [6] précise les conditions pour que l'étape "Away step" soit possible et permet de déduire le lemme suivant:

II.6.3 Lemme:

Soit $\bar{\bar{x}}$ la solution au sous-problème

(II.6.3.1) $\underset{x}{\text{Max}}\ \nabla f(\hat{x})\ x$

sujet à

- $x \in \underline{X}$
- $x_j = 0$ si $\hat{x}_j = 0$
  $x_j \geq 0$ sinon

et λ^* tel que

$$\lambda^* = \text{Max}\ \{(\lambda\ \hat{x} + (1-\lambda)\ \bar{\bar{x}} \in \underline{X}\} > 1\ ;$$

alors en définissant

$$y = \lambda^*\ \hat{x} + (1-\lambda^*)\ \bar{\bar{x}}$$

la direction $d = (y-\hat{x})$ est une direction de descente.

Preuve

Dans un premier temps on doit vérifier que

$$\nabla f(\hat{x})\ (y-\hat{x}) < 0.$$

En développant l'expression de $(y-\hat{x})$ on obtient

$$(\lambda^*-1)\ (\hat{x}-\bar{\bar{x}})$$

Donc,

$$\nabla f(\hat{x})\ (y-\hat{x}) = \nabla f(\hat{x})\ (\lambda^*-1)\ (\hat{x}-\bar{\bar{x}}) = (\lambda^*-1)\ [\nabla f(\hat{x}) - \nabla f(\hat{x})\ \bar{\bar{x}}]$$

Mais puisque

. $\lambda^* > 1 \Rightarrow (\lambda^*-1) > 0$

. $\nabla f(\hat{x})\ \bar{\bar{x}} = \underset{x\in\underline{X}}{\text{Max}}\ [\nabla f(\hat{x})\ x]$

on a alors

$$\nabla f(\hat{x})\ (y-\hat{x}) < 0.$$

Maintenant pour juger de la nécessité des contraintes il faut se référer à la méthode des directions réalisables et au problème (3) dans la section I.2, où $d = (y - \hat{x})$

Les contraintes sont linéaires et sous forme d'égalités, $\underline{X}$ peut s'exprimer sous la forme $Ax = b$. Donc $\hat{x}$, $y \in \underline{X}$ implique $A\hat{x} = b$. D'où la contrainte

$$A\ (y-\hat{x}) = 0.$$

De plus, on doit avoir $d = (y-\hat{x}) \geq 0$. Par conséquent

$$(y-\hat{x}) = (\lambda^*-1)\ \hat{x} + (1-\lambda^*)\ \bar{\bar{x}} \geq 0.$$

ou encore, pour chaque composante j

$$(\lambda^*-1)\ \hat{x}_j + (1-\lambda^*)\ \bar{\bar{x}}_j$$

Et si $\hat{x}_j = 0$ alors

$$(1-\lambda^*)\ \bar{\bar{x}}_j \geq 0,$$

On a donc nécessairement $\bar{\bar{x}}_j = 0$ (puisque $1-\lambda^* < 0$).

Nous sommes maintenant en mesure de décrire les trois directions de descente utilisées dans un nouvel algorithme. Les types de direction sont:

(1) Toward step régulier:

Il s'agit de calculer $x^{\ell+1} = x^{\ell} + t^{\ell+1}(\bar{x}^{\ell+1} - x^{\ell})$ à chaque itération ℓ, où $\bar{x}^{\ell+1}$ est solution du sous-problème, minimiser $\nabla f(x^{\ell})(x - x^{\ell})$ sur $x \in \underline{X}$ exactement comme dans l'algorithme original de Frank et Wolfe qui se trouve à la section I.3 (sans restriction).

(2) Toward step restrictif:

On procède de même sauf en ce qui a trait à $\bar{x}^{\ell+1}$: le sous-problème revient à minimiser $\nabla f(x^{\ell})(x - x^{\ell})$ sur $x \in \{x^j, j \in P\}$, c'est-à dire en se restreignant à cet ensemble. On doit mettre à jour les x^j.

(3) Away step restrictif:

On calcule $x^{\ell+1} = x^{\ell} + t^{\ell+1}(\bar{x}^{\ell+1} - x^{\ell})$ à chaque itération ℓ, où $\bar{x}^{\ell+1}$ correspond à y que l'on retrouve dans le lemme II.6.3; cependant, pour le sous-problème, on restreint $\bar{\bar{x}}$ non à $\underline{X}$ mais plutôt à $\{x^j, j \in P\}$. On doit mettre à jour les α_j.

Remarque:

Dans la section II.5 on a mentionné, dans le cas de l'algorithme général basé sur la restriction et la linéarisation interne, le fait que la résolution du problème (6) se compliquait. En effet on doit résoudre:

$$\begin{aligned} &\underset{\{\alpha\}}{\text{Min}} \quad f(\alpha X) \\ &\text{s.a.} \quad \sum_{j=0}^{L-1} \alpha_j = 1 \\ &\qquad \alpha_j \geq 0 \quad j \in P \\ &\qquad \alpha_j = 0 \quad j \notin P \end{aligned}$$

L'algorithme de Frank et Wolfe limite P à deux éléments et le problème (6) devient:

$$\underset{\{\alpha_1,\alpha_2\}}{\text{Min}} \quad f(x^{\ell+1}) = f(\alpha_1 \bar{x}^{\ell+1} + \alpha_2 x^{\ell})$$

s.a.

$$\alpha_1 + \alpha_2 = 1$$
$$\alpha_1 \geq 0$$
$$\alpha_2 \geq 0$$

En posant $t^{\ell+1} = \alpha_1$ et $(1-t^{\ell+1}) = \alpha_2$ on obtient $x^{\ell+1} = (x^{\ell} + t^{\ell+1}).(\bar{x}^{\ell+1} - x^{\ell})$. Les trois types de direction de descente choisis ramènent le problème (6) à cette forme plus simple.

Toutefois, il serait intéressant de développer une méthode efficace pour résoudre le problème (6) afin d'utiliser l'algorithme général de la section II.5; ce domaine est ouvert à la recherche.

II.7 Algorithme basé sur plusieurs directions de descente

II.7.1 Algorithme:

Etape 0: Accumuler L points de base $\{x^j, j\in P\}$ où $P = \{0,1,\ldots, L-1\}$ selon le "Toward step régulier".

Etape 1: Tant qu'un critère d'arrêt n'est pas satisfait, effectuer des itérations "Toward step restrictif" et "Away step restrictif".

Etape 2: Ajouter un autre point de base à l'ensemble $\{x^j, j\in P\}$ vérifier le critère d'arrêt. S'il n'est pas satisfait on retourne à l'étape 1 sinon on s'arrête.

Le nombre de points de base de la linéarisation interne initiale est ar-

bitraire et on doit se référer à l'expérience et à des connaissances à priori sur la nature du problème.

Le critère d'arrêt, à l'étape 1, peut se baser sur l'analyse de l'expression $|\nabla f(x^{\ell})(\bar{x}^{\ell} - x^{\ell-1})|$, représentant une diminution potentielle de la valeur de la fonction économique; cependant, il est préférable de se baser sur la diminution potentielle d'un "toward step restrictif" plutôt que sur la diminution potentielle d'un "away step restrictif". En effet on constate en pratique, que l'ordre de grandeur de ces diminutions potentielles varie selon le type d'itération. Donc on reprendra l'étape 1 jusqu'à ce que la diminution potentielle d'un "toward step restrictif" devienne insignifiante par rapport à la valeur actuelle de la fonction économique.

Enfin, à l'étape 2, on peut analyser l'amélioration potentielle ou déterminer une borne inférieure à la valeur de la fonction économique.

Les résultats d'un tel algorithme sont sensibles aux critères d'arrêt et de choix entre les différents types d'itération. Ceci sera discuté lors de l'analyse des résultats et de la critique du critère d'arrêt.

CHAPITRE TROISIEME

MODELE DE RESEAU POUR PLUSIEURS PRODUITS A COUT MINIMAL

III.1 Définitions et formulation

En général, le problème d'affectation sur un modèle de réseau à plusieurs produits se conçoit ainsi: étant donné un réseau contenant des centroïdes, des demandes en déplacements de centroïdes vers d'autres centroïdes, une fonction de coût du volume de circulation d'un noeud à un autre, il s'agit de minimiser le coût total de circulation dans le réseau sous des contraintes de conservation de flot (au niveau de la satisfaction des demandes aux centroïdes et au niveau de la circulation aux noeuds intermédiaires).

Le problème d'affectation de trafic choisi est un cas particulier. Etant donné une ville avec des rues et des zones (centroïdes) on estime les déplacements entre ces zones (correspondant à un ensemble de demandes fixes). On cherche à savoir comment sera distribué le trafic dans la ville. De plus, on fait l'hypothèse que les voyageurs ont tous la même perception des coûts et qu'ils cherchent à minimiser leurs coûts. Il survient un problème pratique pour l'évaluation des coûts. Pour plus d'information consulter LeBlanc [15], Nguyen [17] et Potts-Oliver [21].

Il existe deux formulations: arc-chemin et arc-noeud. Nous retiendrons la première (voir Florian [6]) pour l'identification de ce modèle au modèle général développé précédemment; on retrouve un développement de la seconde formulation dans LeBlanc [15].

Soient les définitions:

N	l'ensemble des noeuds:
m	le nombre total de noeuds.
A	l'ensemble des arcs.
p	une origine ($p \in N$).
q	une destination ($q \in N$).
(pq)	un couple origine-destination.
$K_{,(pq)}$	l'ensemble de tous les chemins de l'origine p vers la destination q.
$k_{,(pq)}$	un chemin en particulier $k_{,(pq)} \in K_{(pq)}$
$g_{(pq)}$	la demande en déplacements de l'origine p vers la destination q.
$\delta_{ak,(pq)}$	une fonction indicatrice prenant la valeur 1 si l'arc a est sur le chemin $k_{,(pq)}$ de l'origine p vers la destination q, et 0 sinon.
$h_{k,(pq)}$	la quantité de déplacements sur le chemin $k_{,(pq)}$ en vue de satisfaire la demande $g_{(pq)}$
v_a	la quantité totale de déplacements sur l'arc a (fonction de tous les chemins, de toutes les demandes pour les couples origine-destination).
$s_a(v_a)$	une fonction de la quantité de déplacements sur l'arc a, convexe, monotone non-décroissante et continue.

On retrouve à l'annexe A un exemple de formulation. Le modèle mathématique est:

(III.1.1)

$$(A)\quad \underset{\{h_{k,(pq)}\}}{\text{Minimiser}} \sum_a \int_0^{v_a} s_a(v_a)\,dv_a \qquad \left(=\sum_a S_a(v_a)\right)$$

sujet à:

$$(B)\quad \sum_k h_{k,(pq)} = g_{(pq)} \qquad \text{pour tout } (pq)$$

$$(C)\quad v_a = \sum_h \sum_p \sum_q \delta_{ak,(pq)} \cdot h_{k,(pq)} \qquad \text{pour tout arc } a$$

$$(D)\quad h_{k,(pq)} \geq 0 \qquad \text{pour tout } k,(pq)$$

III.2 Identification au problème général

Dans les sections qui précèdent on retrouve le développement d'un algorithme s'inspirant du "gradient et interpolation" (ainsi surnommé par Frank et Wolfe), de la linéarisation interne du domaine formé par des contraintes linéaires et de la stratégie de restriction.

Cet algorithme s'applique à une fonction $f(x)$: $E^n \to E^1$ convexe et différentiable sur un ensemble convexe $\underline{X}$ et solutionne le problème

$$\underset{x \geq 0,\ x \in \underline{X}}{\text{Minimiser}} f(x)$$

Nous allons montrer que la fonction (A) est convexe et différentiable, que le domaine (B), (C), (D) forme un ensemble convexe et que le sous-problème II.4.2 se réduit à un calcul des chemins les plus courts. De plus, on utilisera une heuristique pour effectuer l'étape "Away step". Enfin on développera l'algorithme II.7.1 pour notre modèle particulier.

III.2.1 Lemme:

Si les fonctions $S_a(v_a)$ sont continûment différentiables et non décroissantes sur l'interval semi-ouvert $[0, \infty)$ alors la fonction économique (A) est convexe par rapport aux variables $h_{k,(pq)}$

Preuve:

Montrons d'abord que $S_a(v_a)$ est une fonction convexe de v_a. On a $S_a'(v_a) = s_a(v_a)$. Puisque $s_a(v_a)$ est, par hypothèse (voir section III.1) continûment différentiable et non décroissante, alors $s_a'(v_a)$ est non négatif sur $[0,\infty)$. Par conséquent, en se rappelant que $S_a(v_a)$ est convexe si et seulement si sa dérivée seconde est non négative sur son domaine (Ferland [5]) la conclusion suit.

L'expression (A) est une combinaison (positive) linéaire de fonctions convexes, donc (A) est convexe par rapport aux v_a (Lasdon [14, p.72].

Enfin, v_a est une combinaison linéaire des $h_{k,(pq)}$. La fonction v_a est convexe par rapport aux variables $h_{k,(pq)}$ et par conséquent, chacune des fonctions composées $S_a(v_a)$ est convexe par rapport aux variables $h_{k,(pq)}$, et le lemme est prouvé.

Sous ces hypothèses, une solution optimale locale au problème (A), (B), (C), (D) est aussi une solution optimale globale. De plus, si on ajou-

te l'hypothèse que les fonctions $S_a(v_a)$ sont strictement croissantes, alors la fonction économique (A) est strictement convexe par rapport aux variables v_a mais non strictement convexe par rapport aux variables $h_{k,(pq)}$. A l'optimum, la répartition des déplacements est unique dans le cas où l'on impose aux fonctions $S_a(v_a)$ d'être strictement croissantes.

III.2.2 Lemme:

L'ensemble des points réalisables pour le système d'équations (B), (C), (D) forme un ensemble convexe.

Preuve:

Soient deux solutions réalisables:

- $h^1_{k,(pq)}$ pour tout $k,(pq)$
- $h^2_{k,(pq)}$ pour tout $k,(pq)$

alors,

pour une combinaison convexe de ces deux solutions réalisables,

$$h^3_{k,(pq)} = \lambda\, h^1_{k,(pq)} + (1-\lambda)\, h^2_{k,(pq)}$$

pour tout $k,(pq), 0 \leq \lambda \leq 1$

on a:

$$(1)\quad \sum_k h^3_{k,(pq)} = \lambda \sum_k h^1_{k,(pq)} + (1-\lambda) \sum_k h^2_{k,(pq)}$$

$$= \lambda\, g_{(pq)} + (1-\lambda)\, g_{(pq)}$$

$$= g_{(pq)}$$

pour tout (pq)

(2) $v_a^3 = \lambda\, v_a^1 + (1-\lambda)\, v_a^2$

regroupement pour tout arc a.

(3) $h^3_{k,(pq)} = \lambda\, h^1_{k,(pq)} + (1-\lambda)\, h^2_{k,(pq)} \geq 0$

pour tout k,(pq)

Donc cette solution est aussi réalisable.

Remarque:

En pratique il s'agit de s'assurer qu'il existe un chemin entre chaque centroïde où il y a une demande en déplacements, puisqu'il n'y a pas de limite de capacité sur les chemins.

Résolution du sous-problème (II.4.2):

Considérons le problème consistant à trouver une direction de descente par interporation linéaire, étant donné une solution $\hat{x}$:

$$\underset{x\in\overline{X}}{\text{Minimiser}} \quad \nabla f(\hat{x})\,(x-\hat{x})$$

De façon équivalente, on peut se limiter à trouver $\bar{x}$ qui minimise $\nabla f(\hat{x})\, x$ (voir section I.3). Pour notre modèle de réseau, on cherche la solution du sous-problème:

$$\text{(III.2.3)} \quad \underset{\{h_{k,(pq)}\}}{\text{Minimiser}} \sum_h \sum_p \sum_q \frac{\partial\left(\sum_a S_a(\hat{v}_a)\right)}{\partial\, h_{k,(pq)}} \cdot h_{k,(pq)}$$

sujet aux contraintes:

(*) $\sum_k h_{k,(pq)} = g_{(pq)}$ pour tout (pq)

(*) $h_{k,(pq)} \geq 0$ pour tout k,(pq)

où,

$$(**)\ \hat{v}_a = \sum_k \sum_p \sum_q \delta_{ak,(pq)} \cdot \hat{h}_{k,(pq)} \qquad \text{pour tout a}$$

$$(**)\ \sum_k \hat{h}_{k,(pq)} = g_{(pq)} \qquad \text{pour tout (pq)}$$

$$(**)\ \hat{h}_{k,(pq)} \geq 0 \qquad \text{pour tout k,(pq)}$$

En développant l'équation de la fonction économique et en utilisant la règle en chaîne, on obtient:

$$(III.2.4)\quad \sum_k \sum_p \sum_q \left[\sum_a s_a(\hat{v}_a) \cdot \delta_{ak,(pq)}\right] \cdot h_{k,(pq)}$$

Le terme $[\sum_a s_a(\hat{v}_a) \cdot \delta_{ak,(pq)}]$ représente la longueur du chemin k,(pq) partant du centroïde p et allant au centroïde pour la solution $\{\hat{h}_{k,(pq)}\}$ et la variable $h_{k,(pq)}$ la quantité de déplacements sur le chemin k,(pq).

Donc, notre sous-problème revient à la satisfaction des demandes en déplacements entre chaque couple origine-destination (pq) en répartissant les déplacements sur les chemins k,(pq) et ce, à moindre coût.

Bien plus, on peut séparer le sous-problème (III.2.3) en problèmes indépendants, en considérant séparément chaque couple origine-destination:

$$(III.2.5)\quad \underset{\{h_{k,(pq)}\}}{\text{Minimiser}}\{\sum_a \sum_k s_a(\hat{v}_a) \cdot \delta_{ak,(11)} \cdot h_{k,(11)} + \ldots$$

$$\ldots + \sum_a \sum_k s_a(\hat{v}_a) \cdot \hat{\delta}_{ak,(pq)} \cdot h_{k,(pq)} \ldots\}$$

sujet à

$$\sum_k h_{k,(11)} = g_{(11)}$$

$$h_{k,(11)} \geq 0$$

$$\sum_k h_{k,(pq)} = g_{(pq)}$$

$$h_{k,(pq)} \geq 0$$

Notations:

On peut expliciter la notation en définissant les ensembles Q et P tels que $q \in Q$, $p \in P$ et les ensembles $K_{,(pq)}$ tels que $k_{,(pq)} \in K_{,(pq)}$

En utilisant le symbole | | pour désigner le nombre d'éléments dans un ensemble, on a alors:

(1) $\sum_k h_{k,(pq)} \equiv \sum_{k \in K_{,(pq)}} h_{k,(pq)}$

(2) $|Q| * |P|$ problèmes indépendants.

(3) $|K_{,(pq)}|$ variables pour le problème indépendant défini par l'origine p et la destination q.

Minimiser la valeur de la somme des problèmes indépendants revient à minimiser chacune des valeurs des fonctions économiques des problèmes. LeBlanc [15] montre pour une formulation particulière basée sur le modèle arc-noeud, que résoudre chaque problème revient à effectuer tous les déplacements sur le chemin $h_{k^*,(pq)}$ tel que $k^*_{,(pq)} \in K_{,(pq)}$ et ce chemin est le plus court.

III.2.6 Théoreme:

A chaque itération de l'algorithme de Frank et Wolfe, la solution optimale du sous-problème (III.2.3) (ou encore III.2.5) est donnée par:

$$\bar{h}_{k,(pq)} = \begin{cases} g_{(pq)} & \text{si le chemin } \bar{k},(pq) \text{ est le plus court chemin entre l'origine p et la destination q, } \bar{k},(pq) \in K,(pq) \\ 0 & \text{pour tout chemin } k,(pq) \in K,(pq) \text{ tel que } k,(pq) \neq \bar{k},(pq). \end{cases}$$

pour tout couple origine-destination (pq).

Preuve:

Supposons que l'on soit à l'optimum pour le sous-problème (III.2.5) et qu'il existe au moins un couple origine-destination tel que tous les déplacements ne soient pas répartis uniquement sur le chemin le plus court k*,(pq) entre p et q.

On a alors une solution $\bar{h}_{k,(pq)}$ telle que:

(1) $\sum_k \bar{h}_{k,(pq)} = g_{(pq)}$

(2) $\bar{h}_{k,(pq)} \neq g_{(pq)}$ pour tout $k,(pq)$

Considérons la solution consistant à répartir sur le chemin k*,(pq) tous les déplacements. On a:

(3) $h^*_{k,(pq)} = g_{(pq)}$ $\quad k,(pq) = k^*,(pq)$

(4) $h^*_{k,(pq)} = 0 \qquad k,(pq) \in K,(pq)$

$k,(pq) \neq k^*_{;}(pq)$

Par définition, on peut écrire:

(5) $\sum_a s_a(\hat{v}_a) \cdot \hat{\delta}_{ak^*_{;}(pq)} \leq \sum_a s_a(\hat{v}_a) \cdot \hat{\delta}_{ak,(pq)}$

pour tout chemin $k,(pq) \in K,(pq)$

Donc

$$\sum_a \sum_k s_a(\hat{v}_a) \cdot \hat{\delta}_{ak,(pq)} \cdot \bar{h}_{k,(pq)}$$

$$\geq \sum_k \sum_a s_a(\hat{v}_a) \cdot \hat{\delta}_{ak^*_{;}(pq)} \cdot \bar{h}_{k,(pq)}$$

$$\geq \left(\sum_a s_a(\hat{v}_a) \cdot \hat{\delta}_{ak^*_{;}(pq)}\right) \cdot \left(\sum_k \bar{h}_{k,(pq)}\right)$$

$$\geq \left(\sum_a s_a(\hat{v}_a) \cdot \hat{\delta}_{ak^*_{;}(pq)}\right) \cdot \left(g_{(pq)}\right)$$

$$\geq \left(\sum_a s_a(\hat{v}_a) \cdot \hat{\delta}_{ak^*_{;}(pq)}\right) \cdot h^*_{k^*_{;}(pq)}$$

$$\geq \sum_a \sum_k s_a(\hat{v}_a) \cdot \hat{\delta}_{ak,(pq)} \cdot h^*_{k,(pq)}$$

ce qui contredit l'hypothèse selon laquelle l'expression:

(III.2.6) $\sum_a \sum_k s_a(\hat{v}_a) \cdot \hat{\delta}_{ak,(pq)} \cdot \bar{h}_{k,(pq)}$

est minimale.

Ainsi, pour le couple origine-destination (pq), on doit avoir

$\bar{h}_{k,(pq)} = h^*_{k,(pq)}$ pour tout $k,(pq) \in K,(pq)$

Enfin, puisque la fonction économique du sous-problème (III.2.5) est formée d'une combinaison linéaire des fonctions du type

(III.2.6) définies sur des domaines indépendants pour chaque couple origine-destination (pq), on doit avoir

$$\bar{h}_{k,(pq)} = h^*_{k,(pq)}$$

pour tout couple origine-destination (pq), ce qui complète la preuve.

Résolution du sous-problème (II.6.3.1):

On doit maintenant résoudre le sous-problème

$$\text{Maximiser}_{x \in \bar{X}} \quad \nabla f(\hat{x})\,(x-\hat{x})$$
$$\hat{x}_j = 0 \Rightarrow x_j = 0$$

Lorsqu'on revient au modèle de réseau, on obtient un sous-problème semblable au sous-problème (III.2.5) sauf en ce qui concerne la fonction économique: on doit maximiser au lieu de minimiser.

Par conséquent, on peut reprendre le même développement que pour le cas du sous-problème (III.2.5) et trouver ce maximum en tenant compte de la solution précédente $\hat{x}$.

Définissions des nouvelles variables $r_{k,(pq)}$ telles que

$$r_{k,(pq)} = \begin{cases} = 0 & \text{si } \hat{h}_{k,(pq)} = 0 \\ \geq 0 & \text{si } \hat{h}_{k,(pq)} > 0 \end{cases}$$

En fait résoudre le sous-problème en question revient à chercher les chemins les plus longs passant par les arcs sur lesquels il y a déjà des

déplacements. Avec le changement de variable on obtient:

$$(III.2.7)\quad \underset{\{r_{k,(pq)}\}}{\text{Maximiser}} \{ \sum_a \sum_k s_a(\hat{v}_a) \cdot \hat{\delta}_{ak,(11)} \cdot r_{k,(11)} + \ldots$$

$$\ldots + \sum_a \sum_k s_a(\hat{v}_a) \cdot \hat{\delta}_{ak,(pq)} + \ldots \}$$

sujet à:

$$\sum_k r_{k,(11)} = g_{(11)}$$

$$r_{k,(11)} = \begin{cases} = 0 & \text{si } \hat{h}_{k,(11)} = 0 \\ \geq 0 & \text{si } \hat{h}_{k,(pq)} > 0 \end{cases}$$

$$\sum_k r_{k,(pq)} = g_{(pq)}$$

$$r_{k,(pq)} = \begin{cases} = 0 & \text{si } \hat{h}_{k,(pq)} = 0 \\ \geq 0 & \text{si } \hat{h}_{k,(pq)} > 0 \end{cases}$$

On peut démontrer le théorème suivant:

III.2.8 Théorème:

A chaque itération de l'algorithme de Wolfe (II.6.1), la solution optimale du sous-problème (III.2.7) est donnée par:

$$\bar{r}_{k,(pq)} = \begin{cases} g_{(pq)} & \text{si le chemin } k^*,(pq) \text{ est le plus long de l'origine p vers la destination q passant par les arcs sur lesquels il y a déjà des déplacements.} \\ 0 & \text{sinon} \end{cases}$$

La preuve du théorème précédent est omise. En se référant à la preuve du théorème III.2.6 on reprend le même raisonnement en inversant le signe des inégalités. On fait l'hypothèse que les chemins sont définis de telle sorte qu'ils ne comportent pas de cycle positif (cycle dans lequel chaque passage augmente la longueur du chemin).

En pratique il est très difficile de construire un algorithme permettant d'exclure les chemins comportant des cycles positifs. C'est pourquoi nous utiliserons une heuristique consommant moins de temps de calcul, basée sur la stratégie de restriction. Ainsi nous pourrons compléter, dans les limites de temps raisonnables, les calculs nécessaires afin de choisir entre les directions de descente possibles et utiliser l'algorithme (II.7.1).

Interprétation intuitive

A chaque itération "ℓ" de la méthode d'approximation linéaire, étant donné une solution en main $\{h^{\ell}_{k,(pq)}\}$, on résoud le sous problème (III.2.3) consistant à trouver les chemins les plus courts pour chaque couple origine destination (en fait on trouve une nouvelle direction améliorant la valeur de la fonction économique par rapport à la solution courante); puis on transfert des déplacements, appartenant à cette solution en main, sur les chemins les plus courts, diminuant ainsi la valeur de la fonction économique, tout en restant dans les limites possibles (il ne faut

pas transférer plus de déplacement qu'il n'en existe). C'est ce que Wolfe [23] appelle se déplacer à travers les solutions en main ("Toward step"). Comme il le fait remarquer, les solutions obtenues au cours des itérations antérieures ont un poids sur la solution que l'on obtiendra.

D'autre part, lorsqu'on résoud, à l'itération "ℓ", le sous-problème (III.2.7), on trouve les chemins les plus longs pour tout couple origine-destination; puis on retire des déplacements appartenant à la solution courante, sur les chemins les plus longs pour lesquels il y a circulation, tout en gardant la nouvelle solution réalisable. Donc on s'éloigne de la solution en main ("Away step").

En combinant ces deux méthodes d'amélioration de la valeur de la fonction économique on obtient une plus grande richesse au niveau des possibilités de directions de convergence vers l'optimum.

III.3 Stratégie de restriction:

On désigne le numéro d'itération par la variable "ℓ". Pour faciliter le développement de la théorie, on introduit la notation:

$$(III.3.1)\quad v_a^{\ell} = \sum_k \sum_p \sum_q \delta_{ak,(pq)}^{\ell} \cdot h_{k,(pq)}^{\ell} \qquad \text{pour tout arc a,}$$

représentant la quantité de déplacements.

En utilisant la linéarisation interne du domaine (polytope) on peut exprimer la solution courante v_a^{ℓ} en fonction de solutions antérieures v_a^{j}, $j=0,1,\ldots \ell$. Ces solutions antérieures $\bar{v}_a^{j}$ sont les solutions optimales du sous-problème (III.2.3) à l'itération j.

Dans le problème général, à chaque itération ℓ, après avoir choisi une direction de descente $d = (\bar{x}^{\ell} - x^{\ell-1})$, on trouve t^{ℓ}, tel que

$$f(x^{\ell-1}+t^{\ell}(\bar{x}^{\ell} - x^{\ell-1}))$$

est minimisé (sur le domaine), jusqu'à ce qu'un critère d'arrêt soit satisfait.

Pour notre fonction économique (III.1.1) (A), on cherche t^{ℓ}, tel que

$$\sum_a S_a \, (v_a^{\ell-1}+t^{\ell}(\bar{v}_a^{\ell} - v_a^{\ell-1}))$$

soit minimum (sur le domaine), jusqu'à ce qu'un critère d'arrêt soit satisfait. Sachant que $0 \leq t^{\ell} \leq 1$, on peut établir un lemme de base pour la stratégie de restriction.

III.3.2 Lemme:

La solution courante, à l'itération ℓ, peut s'écrire comme combinaison linéaire convexe des points extrémaux $\bar{v}_a^{\,j}$ $j = 0,\ldots,\ell$, solutions optimales du sous-problème (III.2.3) à l'itération j, pour chaque arc a; les équations mathématiques sont:

$$v_a^{\ell} = \sum_{j=0}^{\ell} \alpha_j^{\ell} \bar{v}_a^{\,j}$$

où $\sum_{j=0}^{\ell} \alpha_j^{\ell} = 1$

$$\alpha_j^{\ell} \geq 0 \qquad j = 0,\ldots,\ell$$

Preuve:

A l'itération initiale on a:

$$v_a^0 = \bar{v}_a^0 = \alpha_0^0 \, v_a^0$$

où , $\alpha_0^0 = 1$.

A la fin de l'itération 1, on a:

$$v_a^1 = v_a^0 + t^0 \, (\bar{v}_a^1 - v_a^0)$$

$$= (1-t^1) \, v_a^0 + (t^1) \, \bar{v}_a^1$$

$$= (\alpha_0^1) \, v_a^0 + (\alpha_1^1) \, \bar{v}_a^1$$

A la fin de l'itération ℓ, on a:

$$v_a^\ell = v_a^{\ell-1} + t^\ell \, (\bar{v}_a^\ell - v_a^{\ell-1})$$

$$= (1-t^\ell) \, v_a^{\ell-1} + (t^\ell) \, \bar{v}_a^\ell \quad .$$

En substituant $v_a^{\ell-1}$ par sa valeur exprimée en fonction de $\bar{v}_a^{\ell-1}$ et $\bar{v}_a^{\ell-2}$, on obtient:

$$v_a^\ell = (1-t^\ell) \, (1-t^{\ell-1}) \, \bar{v}_a^{\ell-2} + (1-t^\ell) \, (t^{\ell-1}) \, \bar{v}_a^{\ell-1} + (t^\ell) \, \bar{v}_a^\ell \quad .$$

On substitue ainsi les v_a^j par leur valeur jusqu'à ce qu'on obtienne v_a^ℓ en fonction des $\bar{v}_a^j$:

$$v_a^\ell = \left[\sum_{j=0}^{\ell-1} \prod_{i=j+1}^{\ell} (1-t^i) \, \bar{v}_a^j \right] + t^\ell \, \bar{v}_a^\ell \quad .$$

En posant

$$\cdot\ \alpha_{\ell}^{\ell} = t^{\ell}$$

$$\cdot\ \alpha_{j}^{\ell} = \prod_{i=j+1}^{\ell} (1-t^{i}) \qquad j = 0,\ldots,\ell-1$$

on obtient:

$$v_a^{\ell} = \sum_{j=0}^{\ell} \alpha_j^{\ell}\, \bar{v}_a^{j} \ .$$

Pour vérifier que les contraintes sont respectées, il suffit de décomposer successivement l'expression:

$$\sum_{j=0}^{\ell} \alpha_j^{\ell} \ .$$

On obtient:

$$\cdot\Big[\sum_{j=0}^{\ell-1} \prod_{i=j+1}^{\ell} (1-t^{i}) + t^{\ell}\Big] = (1-t^{\ell})\Big[\sum_{j=0}^{\ell-2} \prod_{i=j+1}^{\ell-1} (1-t^{i})\Big] + (t^{\ell})$$

$$\cdot\Big[\sum_{j=0}^{\ell-2} \prod_{i=j+1}^{\ell-1} (1-t^{i})\Big] = (1-t^{\ell-1})\Big[\sum_{j=0}^{\ell-3} \prod_{i=j+1}^{\ell-2} (1-t^{i})\Big] + (t^{\ell-1})$$

où $0 \le t^{\ell-1} \le 1$

$$\cdot\big[(1-t^{1}) + t^{1}\big] = 1$$

Donc chacune des expressions entre crochet est égale à 1. La valeur de t^{ℓ} est choisie de telle façon que ces résultats restent valables pour tout arc a.

Remarque:

La règle récursive de mise à jour des valeurs des α_j^ℓ à l'itération ℓ s'énonce:

$$\cdot\ \alpha_j^\ell = \alpha_j^{\ell-1}\ (1-t^\ell) \qquad j = 0,\ldots,\ell-1$$

$$\cdot\ \alpha_\ell^\ell = t^\ell$$

Dans la section (II.6) on a noté que l'algorithme de Frank et Wolfe limite l'ensemble P à deux éléments; en tenant compte du lemme précédent, le problème (6) devient (en effectuant les changements de notations appropriés):

$$\underset{\{t^{\ell+1}\}}{\text{Min}}\ f(x^{\ell+1}) = f(x^\ell + t^{\ell+1}(\bar{x}^{\ell+1} - x^\ell))$$

$$0 \le t^{\ell+1} \le 1$$

et en substituant $\bar{x}^\ell$ par une expression analogue à celle du lemme précédent:

$$\underset{\{t^{\ell+1}\}}{\text{Min}}\ f([\sum_{j=0}^{\ell} \alpha_j^\ell\, \bar{x}^j] + t^{\ell+1}(\bar{x}^{\ell+1} + [\sum_{j=0}^{\ell} \alpha_j^\ell\, \bar{x}^j]))$$

$$0 \le t^{\ell+1} \le 1\ .$$

Donc minimiser $f(x^{\ell+1})$ sur le segment de droite reliant $\bar{x}^{\ell+1}$ et x^ℓ peut s'exprimer en fonction des points extrêmes précédents, solutions des problèmes de la forme (II.4.2) chacun étant associé à un $\bar{x}^j$. On aurait pu développer le lemme III.3.2 pour le modèle général et l'appliquer à notre modèle.

Ainsi, le cas particulier de la stratégie de restriction, correspondant

à la stratégie de l'algorithme de Frank et Wolfe, peut s'exprimer de façon différente en se basant sur le lemme III.3.2; ceci est nécessaire pour introduire les sections suivantes.

III.4 Adaptation d'un "Away step" restrictif

Il convient maintenant d'introduire l'heuristique nécessaire pour obtenir une direction permettant l' "Away step" sans avoir à résoudre le sous-problème (III.2.7), ou bien, de façon équivalente le sous-problème (III.2.5), lorsqu'on se réfère au modèle général.

En fait, à chaque solution $\bar{v}_a^j$ (pour tout arc a), on fait une affection "tout ou rien" qui consiste à satisfaire les demandes en répartissant tous les déplacements sur les chemins les plus courts entre tout couple origine-destination.

Lorsqu'on veut effectuer un "Away step", on a à l'itération ℓ, un ensemble de coûts $s_a(v_a^{\ell-1})$ associés à chaque arc a; au lieu de résoudre le sous-problème (III.2.7) sur le domaine entier possible pour les variables $\{r_{k,(pq)}\}$ définies en fonction des variables $\{h_{k,(pq)}\}$, on peut restreindre à 0 toutes les variables $\{r_{k,(pq)}\}$, sauf celles définies en fonction des variables $\{h_{k,(pq)}\}$ dont le domaine se trouve parmi les valeurs $\{\bar{h}^j_{k,(pq)}\}$ $j = 0,1,\ldots,L-1$, trouvées lors de l'application de l'algorithme original de Frank et Wolfe. Une autre façon de voir ceci consiste à observer qu'on se limite à composer une direction de descente à partir de L solutions $\{\bar{v}_a^j\}$; l'heuristique vient du fait qu'on ne recherche plus les chemins les plus longs sur tout le domaine possible,

mais parmi quelques solutions antérieures; cette approche est plausible car, relativement à la solution courante, ces solutions antérieures impliquent lors de la résolution des sous-problèmes, des chemins plus longs que ceux en cours (calculés à partir des coûts $s_a(v_a^{\ell-1})$).

Donc, il s'agit maintenant de chercher l'indice j* donnant la solution au sous-problème:

(III.4.1) Maximiser $\sum_a s_a^{\ell-1}(v_a^{\ell-1}) \cdot \bar{v}_a^j$
$0 \le j \le L-1$

On peut énoncer le lemme suivant:

III.4.2 Lemme:

Soit j* l'indice pour lequel $\bar{v}_a^{j*}$ résoud le sous-problème (III.4.1). On obtient une direction réalisable permettant une diminution de la valeur de la fonction économique en calculant

$$d_a = (\lambda^*-1)(v_a^{\ell-1} - \bar{v}_a^{j*}) \qquad \text{pour tout arc a}$$

où

$$\lambda^* = (1-\alpha_{j*})^{-1} .$$

De plus, la mise à jour des coefficients de la combinaison linéaire convexe des points extrémaux

$$v_a^{\ell} = \sum_{j=0}^{L-1} \alpha_j^{\ell} \bar{v}_a^j$$

s'effectue par la règle:

(III.4.3) . $\alpha_j^{\ell} = \alpha_j^{\ell-1}(1+t^{\ell}\lambda^*-\lambda^*)$ pour tout j, $j \neq j^*$

. $\alpha_j^{\ell} = \alpha_{j^*}^{\ell-1}(1+t^{\ell}\lambda^*-\lambda^*) + t^{\ell}(1-\lambda^*)$ pour tout $j = j^*$.

Preuve:

Par le lemme (III.6.3) on a établi la direction de descente pour l' "Away step":

$$(y-\hat{x}) = (\lambda^*-1)\,\hat{x} + (1-\lambda^*)\,\bar{\bar{x}} \geq 0,$$

ou bien de façon équivalente,

$$\lambda^*\,\hat{x}_j + (1-\lambda^*)\,\bar{\bar{x}}_j \geq \hat{x}_j \qquad \text{pour tout } j.$$

Mais puisque $\hat{x}_j \geq 0$, on doit respecter:

$$\lambda^*\,\hat{x}_j + (1-\lambda^*)\,\bar{\bar{x}}_j \geq 0 \qquad \text{pour tout } j.$$

Pour le problème en question, ceci équivaut à

$$\lambda^*\,v_a^{\ell-1} + (1-\lambda^*)\,\bar{v}_a^{j^*} \geq 0 \qquad \text{pour tout arc a.}$$

En utilisant la décomposition de $v_a^{\ell-1}$ définie par le lemme (III.3.2), tout en se rappelant qu'on se limite à L points extrémaux, on peut récrire:

$$(III.4.4)\quad \lambda^*\Big[\sum_{j=0}^{L-1} \alpha_j^{\ell-1}\,\bar{v}_a^{j}\Big] + (1-\lambda^*)\bar{v}_a^{j^*} \geq 0$$

$$0 \leq j^* \leq L-1.$$

L'affection "tout ou rien" consiste dans ce cas-ci à répartir les déplacements selon le mode:

$$\cdot\ \bar{v}_a^{j} = 0 \qquad\qquad j \neq j^*$$

$$\cdot\ \bar{v}_a^{j^*} = \sum_k \sum_p \sum_q \delta_{ak,(pq)}^{j^*} \cdot h_{k,(pq)}^{j^*}$$

où

$$\cdot\ \sum_k h_{k(pq)}^{j^*} = g_{(pq)} \qquad \text{pour tout } (pq)$$

$$\cdot\ h_{k,(pq)}^{j^*} > 0$$

Donc, l'équation (III.4.4) se réduit à:

$$\lambda^* [\alpha_{j^*} \bar{v}_a^{j^*}] + (1-\lambda^*) \bar{v}_a^{j^*} \geq 0 \qquad \text{pour tout arc a}$$

et $\quad 0 \leq j^* \leq L-1$

d'où on tire la relation:

$$\lambda^* \leq (1-\alpha_{j^*})^{-1} \qquad \text{pour tout arc a.}$$

On retient la valeur de λ^* qui satisfait l'égalité, soit:

$$\underline{\lambda^* = (1-\alpha_{j^*})^{-1}}$$

La direction de descente réalisable devient:

$$d_a = (\lambda^*-1)\, v_a^{\ell-1} + (1-\lambda^*)\, \bar{v}_a^{j^*}$$

$$= (\lambda^*-1)\, (v_a^{\ell-1} - \bar{v}_a^{j^*}) \qquad \text{pour tout arc a.}$$

Pour la règle de mise à jour des α_j^{ℓ}, il s'agit de remarquer que l'on a calculé, à l'itération ℓ, t^{ℓ} tel que l'expression

$$\sum_a S_a\, (v_a^{\ell-1} + t^{\ell} d_a)$$

soit minimale. En substituant d_a par sa valeur, on obtient,

$$v_a^{\ell} = (1+t^{\ell}\ \lambda^*-\lambda^*)\ [v_a^{\ell-1}] + (t^{\ell})\ (1-\lambda^*)\ \bar{v}_a^{j^*}$$

$$= (1+t^{\ell}\ \lambda^*-\lambda^*)\ [\sum_{j=0}^{L-1} \alpha_j^{\ell-1}\ \bar{v}_a^{j}] + (t^{\ell})\ (1-\lambda^*)\ \bar{v}_a^{j^*}$$

$$0 \le j^* \le L-1\ .$$

d'où la règle de mise à jour (III.4.3).

En somme on a conservé L solutions réalisables; on choisit lorsqu'on effectue l'étape "Away step", la solution qui nous coûte le plus cher, étant donné des coûts associés aux arcs pour la solution courante; puis, à partir de cette solution, on forme une direction de descente.

Tous les résultats qui précèdent reposent sur la décomposition introduite dans le lemme (III.3.2). En pratique, on utilise la mémoire auxiliaire pour garder les L solutions et à chaque itération, on recalcule les produits de l'expression (III.4.1) et on met à jour les coefficients α_j^{ℓ} selon la règle (III.4.3).

III.5 Adaptation d'un "Toward step" restrictif

Comme à la section précédente, au lieu de résoudre le sous-problème (III.2.3), on peut se limiter aux L points extrémaux conservés en mémoire auxiliaire et chercher j** tel que $\bar{v}_a^{j^{**}}$ résolve le sous-problème:

(III.5.1) Minimiser $\sum_a s_a^{\ell-1}\ (v_a^{\ell-1})\ .\ \bar{v}_a^{j}$.
$0 \le j \le L-1$

On obtient le lemme:

III.5.2 Lemme:

Soit j** l'indice pour lequel $\bar{v}_a^{j**}$ résoud le sous-problème (III.5.1). Si

$$\sum_a s_a^{\ell-1} (v_a^{\ell-1}) \cdot \bar{v}_a^j < 0$$

alors

$$d_a = (\bar{v}_a^{j**} - v_a^{\ell-1}) \qquad \text{pour tout arc a}$$

est une direction de descente. De plus, la mise à jour des coefficients α_j^ℓ de la combinaison linéaire des points extrémaux:

$$v_a^\ell = \sum_{j=0}^{L-1} \alpha_j^\ell \bar{v}_a^j$$

s'effectue pour

$$\cdot\ \alpha_j^\ell = \alpha_j^{\ell-1} (1-t^\ell) \qquad \text{pour tout } j,\ j \neq j^{**}$$

(III.5.3)

$$\cdot\ \alpha_{j^{**}}^\ell = \alpha_{j^{**}}^{\ell-1} (1-t^\ell) + t^\ell$$

Preuve:

L'ensemble $\{d_a\}$ est une direction de descente par le lemme I.2.2.

Pour la règle de mise à jour:

$$v_a^\ell = v_a^{\ell-1} + t^\ell (\bar{v}_a^{j**} - v_a^{\ell-1})$$

$$= (1-t^\ell) [v_a^{\ell-1}] + (t^\ell) \bar{v}_a^{j**}$$

$$= (1-t^\ell)[\sum_{\substack{j=0 \\ j\neq j^{**}}}^{L-1} \alpha_j^{\ell-1} \bar{v}_a^j] + ((1-t^\ell) \alpha_{j^{**}}^{\ell-1} + t^\ell) \bar{v}_a^{j**}$$

$$0 \leq j^{**} \leq L-1$$

$$= \sum_{j=0}^{L-1} \alpha_j^{\ell} \, \bar{v}_{\alpha}^{j}$$

lorsqu'on symbolise $\{\alpha_j^{\ell}\}$ selon la règle (III.5.3)

CHAPITRE QUATRIEME

ALGORITHME BASE SUR PLUSIEURS DIRECTIONS DE DESCENTE POUR LE MODELE DE RESEAU

L'algorithme qui suit (voir Florian [6]) génère L points extrêmes à partir de la méthode d'approximation linéaire de Frank et Wolfe, puis génère de nouvelles solutions selon trois choix de direction de descente:

(1) Trouver une direction de descente selon la méthode "Away step restrictif" en se limitant aux L solutions conservées; voir la section III.4

(2) Trouver une direction de descente selon la méthode d'approximation (par interpolation linéaire) de la fonction économique évaluée au point désignant la solution courante, en se restreignant aux L solutions conservées; voir la section (III.5) pour le "Toward step restrictif".

(3) Trouver une direction de descente selon la méthode (2) mais sans restriction (Toward step); voir le sous-problème (III.2.3) et la remarque à la section (III.3); dans ce cas on conserve la solution du sous-problème (III.2.3) et on raffine la linéarisation interne du domaine.

Notations:

(a) L'ensemble sur tous les arcs a $\{\bar{v}_a^{\ell}\}$ = EXFLOW $(\{s_a(v_a^{\ell-1})\})$ désigne,

à l'itération ℓ, un nouveau point extrémal solution du sous-problème (III.2.3), étant donné l'ensemble des coûts sur les arcs $\{s_a(v_a^{\ell-1})\}$, fonctions de la solution courante $\{v_a^{\ell-1}\}$.

(b) t^{ℓ} = ONEDIM $(\{S_a(v_a^{\ell-1}, d_a)\})$ désigne, à l'itération ℓ, le calcul du pas dans la direction de descente utilisée dans l'expression:

$$v_a^{\ell} = v_a^{\ell-1} + t^{\ell}\, d_a \qquad \text{(pour tout arc a)}$$

où,

$$0 \le j \le L-1$$

tel que $\{v_a^{\ell}\}$ soit réalisable, et la valeur de la fonction économique soit minimale pour $\{v_a^{\ell}\}$. Ce calcul est fonction de la solution courante $\{v_a^{\ell-1}\}$ et de la direction de descente $\{d_a\}$.

(c) $\{d_a\}$ est une direction de descente obtenue selon l'une des trois méthodes.

(d) ε un nombre petit.

(e) $\{d_a\}^A$, $\{d_a\}^{TR}$, $\{d_a\}^T$ sont des directions de descente obtenues selon les méthodes (1), (2) et (3) respectivement.

Remarque:

Il faut distinguer $\{\bar{v}_a^{\ell}\}$ et $\{v_a^{\ell}\}$. Le premier ensemble désigne une solution optimale du sous-problème (III.2.5) (direction selon Frank et Wolfe) tandis que le second ensemble désigne la solution courante à l'itération ℓ. En se référant au modèle général, on parle de $\bar{x}^{\ell}$ et de x^{ℓ} respectivement.

Algorithme: "version 2"

Etape 0 : [initialisations]

. $\{\bar{v}_a^0\}$ + EXFLOW $(\{s_a(0)\})$;

. garder $(\{\bar{v}_a^0\})$;

. $\alpha_0^0 = 1$;

. $\ell = 0$; (* compteur d'itérations *)

. $u = 1$; (*signale si on a complété l'accumulation de points extrémaux *)

Etape 1 : [calcul de L-1 points extrémaux selon (3)]

. $\ell = \ell + 1$;

. si $(\ell = L)$ alors

$u = 2$

aller à l'étape 3;

Etape 2 : [ajoute un point extrémal]

. $\{\bar{v}_a^\ell\}$ = EXFLOW $(\{s_a(v_a^{\ell-1})\})$:

. garder $(\{\bar{v}_a^\ell\})$;

. $\{d_a\} = \{d_a\}^T = \{\bar{v}_a^\ell - v_a^{\ell-1}\}$

. (* critère d'arrêt *)

si $|\sum_a s_a(v_a^{\ell-1}) \cdot d_a \; / \; \sum_a S_a(v_a^{\ell-1})| \leq \varepsilon$

alors

on termine

(* la solution $\{v_a^{\ell-1}\}$ est acceptable *)

sinon aller à l'étape 7;

Etape 3 : [détermine une direction de descente selon (2)]

. $\ell = \ell + 1$;

. $Z = \sum_a s_a(v_a^{\ell-1}) \cdot \bar{v}_a^{j**} =$

$$\underset{0\le j\le L-1}{\text{Minimum}} \sum_a s_a(v_a^{\ell-1}) \cdot \bar{v}_a^{j} ;$$

. si $(Z < 0)$ alors

$$\{d_a\}^{TR} = \{\bar{v}_a^{j**} - v_a^{\ell-1}\}$$

sinon

$$Z = 0;$$

Etape 4 : [détermine une direction de descente selon (1)]

. $\sum_a s_a(v_a^{\ell-1}) \cdot \bar{v}_a^{j*} = \underset{0\le j\le L-1}{\text{Maximum}} \sum_a s_a(v_a^{\ell-1}) \cdot \bar{v}_a^{j}$;

. $\lambda^* = (1-\alpha_{j*})^{-1}$;

. si $\lambda^* > 1$ alors

$$\{d_a\}^{A} = \{(\lambda^*-1)\ (v_a^{\ell-1} - \bar{v}_a^{j*})\} ;$$

Etape 5 : [alterne entre $\{d_a\}^{TR}$ et $\{d_a\}^{A}$ si possible]

. si (mod $(\ell,2) = 1$) et $(\lambda^* > 1)$ alors

$$\{d_a\} = \{d_a\}^{A}$$

sinon

si $(Z < 0)$ alors

$$\{d_a\} = \{d_a\}^{TR}$$

sinon, aller à l'étape 9;

Etape 6 : [vérifie si une direction de descente basée sur la restriction donne suffisamment d'amélioration.]

si $|\sum_a s_a(v_a^{\ell-1}) \cdot d_a / \sum_a S_a(v_a^{\ell-1})| \leq \epsilon$ alors

$u = 1$,

aller à l'étape 9;

Etape 7 : [calcule t^ℓ optimal et met à jour les α_j selon le cas]

. t^ℓ = ONEDIM ($\{S_a(v_a^{\ell-1}, d_a)\}$;

. $\{v_a^\ell\} = \{v_a^{\ell-1} + t^\ell d_a\}$

. si ($u = 1$) alors

(* direction selon (3) *)

. $\alpha_j^\ell = \alpha_j^{\ell-1} (1-t^\ell)$ $\quad j = 0,\ldots,L-1$

. $\alpha_\ell^\ell = t^\ell$

. si ($u = 2$) alors

si (mod(ℓ,2) = 1) alors

(* direction selon (2) *)

. $\alpha_j^\ell = (\alpha_j^{\ell-1}).(1-t^\ell)$ $\quad j = 0,\ldots,L-1$ et $j \neq j^{**}$

. $\alpha_{j^{**}}^\ell = (\alpha_j^{\ell-1}).(1-t^\ell) + t^\ell$

sinon (*direction selon (1)*)

. $\alpha_j^\ell = (\alpha_j^{\ell-1}) \cdot (1-t^\ell - t^\ell \lambda^*)$ $\quad j=0,\ldots,L-1$ et $j \neq j^*$

. $\alpha_{j*}^{\ell} = (\alpha_{j*}^{\ell-1}).(1-t^{\ell}-t^{\ell}\lambda^*)+t^{\ell}.(1-\lambda^*)$;

Etape 8 : [contrôle si on est dans une phase restrictive (1) ou (2), ou dans une phase de raffinement de la linéarisation interne du domaine (3)]

si ($\ell < L$) alors

aller à l'étape 2

sinon

aller à l'étape 3;

Etape 9 : [génère un point extrême additionnel]

. $\ell = \ell + 1$;

. $L = L + 1$;

. $u = 1$;

. $\{\bar{v}_a^L\} = \text{EXFLOW}\ (\{s_a(v_a^{\ell-1})\})$;

. garder $(\{\bar{v}_a^L\})$;

. $\{d_a\} = \{\bar{v}_a^L - v_a^{\ell-1}\}$;

(* critère d'arrêt *)

si $|\sum_a s_a(v_a^{\ell-1}) \cdot d_a \ / \ \sum_a S_a(v_a^{\ell-1})| \leq \varepsilon$

alors on termine

(* la solution $v_a^{\ell-1}$ est acceptable *).

sinon retourner à l'étape 7.

CHAPITRE CINQUIEME

RESULTATS

V.1 Essais sur des problèmes pratiques

Dans cette section nous désignerons par VERSION-1 la méthode d'approximation linéaire de Frank et Wolfe, codifiée par Nguyen et James [18] et par VERSION-2 le code de l'algorithme que l'on retrouve dans la section IV.

Le premier problème consiste en un réseau de 3 centroïdes, 3 noeuds intermédiaires, 10 arcs (dont 5 ont des coûts associés). Il est reproduit à l'Annexe A et provient de Potts-Oliver [21, page 96]. Pour ce problème, H. Kuhn [11] a obtenu une solution avec une précision de 6 décimales, en utilisant une méthode de point fixe; le code de cet algorithme se nomme Pathfix. Ces résultats servent de point de référence.

On peut comparer les deux versions à l'aide du Tableau I. La VERSION-2 domine l'ancienne sur le plan convergence. On a observé que l'algorithme de Frank et Wolfe permet un taux de convergence rapide au départ, puis se stabilise, apportant des améliorations plus modestes de la valeur de la fonction économique par la suite. La VERSION-2 permet de relancer la convergence en apportant une nouvelle source de directions de descente.

Il est à noter qu'une itération régulière de l'algorithme de la VERSION-2 consiste en un calcul des plus courts chemins ainsi qu'en une recherche itérative d'un minimum d'une fonction d'une seule variable, tandis qu'une itération restrictive consiste en une étape d'éloignement ("Away step") ou à travers le gradient ("Toward step") et en une recherche itérative

TABLEAU I

Conventions: . CPU (Computer processor unit, C.D.C. 173).
. res (itérations restrictives).
. reg (itérations restreintes).

VERSION-1

arc	50 iter.	100 iter.	200 iter.
1	4.74033	4.65424	4.65487
2	1.25233	1.34412	1.34477
3	4.80999	4.69549	4.68711
4	2.55033	2.34973	2.34198
5	4.51200	4.68988	4.68989
Valeur de la fonction économique	9.556585	9.19803	9.170027
CPU (secondes)	.73	1.35	2.631

VERSION-2

arc	3 reg. 49 res.	4 reg. 59 res.	Solution Pathfix
1	4.64284	4.64211	4.642363
2	1.32563	1.32614	1.326307
3	3.69437	4.67386	4.673693
4	2.31721	2.31597	2.316056
5	4.68279	4.68403	4.683944
Valeur de la fonction économique	9.058707	9.058693	9.058692
CPU (secondes)	1.24	1.29	7.48 sec (IBM 360/91)

du minimum d'une fonction d'une seule variable.

Le deuxième problème est un réseau formé de 9 noeuds et 36 arcs, que l'on retrouve dans le livre de Steenbrink [22, page 108]. La valeur de la fonction économique obtenue par le programme Pathfix est 16961. La VERSION-1 donne une solution de 17024 après 50 itérations (4.31 sec. CPU), tandis que pour la VERSION-2, on a 17022 après 5 itérations régulières et 11 itérations restrictives.

Le troisième essai porte sur un problème d'affectation de trafic avec demandes fixes pour la ville de Hull. Le réseau comprend 27 origines/destinations (centroïdes), 155 noeuds au total et 376 arcs. En utilisant l'algorithne de la VERSION-1, on s'aperçoit que le taux de convergence est rapide pour les premières itérations; c'est pourquoi on choisit de se restreindre au départ à 10 points extrêmes (L=9) pour l'algorithme de la VERSION-2 (L augmente lorsqu'on relâche les contraintes). Après 50 itérations de la VERSION-1 (73.2 sec. CPU) la valeur de la fonction économique s'élève à 168,325 tandis que pour la VERSION-2 cette valeur est 167,482 (57.3 sec. CPU) pour 15 itérations régulières et 60 itérations restrictives.

Le dernier problème est aussi un problème d'affectation de trafic avec demandes fixes pour la ville de Winnipeg. Le réseau comporte 147 centroïdes (origines/destinations), 1283 noeuds et 2913 arcs. La solution obtenue à l'aide de la VERSION-1 est de 854,069.91 après 20 itérations; en moyenne chaque itération nécessite 45 secondes de CPU sur un ordina-

teur C.D.C. 173. Pour la VERSION-2, on garde initialement 10 points extrêmaux, et après 16 itérations régulières (points extrêmes) et 40 itérations restrictives, la valeur de la fonction économique s'élève à 845,013.96.

Pour tous les essais qui précèdent on a fixé la tolérance ε à 10^{-3} pour le contrôle de l'amélioration potentielle de la valeur de la fonction économique et ε' à 10^{-4} pour la recherche itérative de la valeur de t^{ℓ}.

On a remarqué, au cours des essais, une instabilité au niveau de la convergence, selon la priorité que l'on accordait aux itérations restrictives, le nombre de points extrêmes conservés, la tolérance ε' au niveau de la recherche itérative de la valeur t^{ℓ}. On observe:

(1) La méthode d'approximation linéaire de Frank et Wolfe possède un taux de convergence rapide au départ puis converge moins rapidement à cause d'une tendance au zig-zag, (voir Zangwill [24, pages 148-162]).

(2) La détermination du nombre initial de points extrêmes à conserver est arbitraire et dépend de l'intuition et du budget de temps de calcul.

(3) On peut augmenter la vitesse de convergence en équilibrant adéquatement le partage entre les itérations restrictives et les itérations régulières; l'équilibre tel que fixé dans la section IV pour la VERSION-2 produit des performances supérieures à celles obtenues par la VERSION-1 et ce pour tous les problèmes que l'on a considérés.

(4) Pour les itérations restrictives, l'amélioration potentielle de la fonction économique pour une direction obtenue à partir d'un "Toward step" est supérieure en valeur absolue, à celle d'une direction obtenue à partir d'un "Away step"; pour tous les problèmes considérés. Donc, on ne peut comparer U1 et U2 conformément à la version de l'algorithme de Wolfe (II.6.1) Dans la VERSION-2 on a choisi d'alterner entre les deux types d'itérations restrictives.

(5) Pour le réseau de la ville de Hull, 34 itérations régulières, 12 "Toward step" et 9 "Away step" conduisent à une solution qui s'élève à 167,349.4; un autre essai répartissant autrement le partage entre les itérations régulières et les itérations restrictives, donne une solution de 167,732.8 pour 34 itérations régulières, 7 "Toward step" et 4 "Away step"; ceci confirme l'instabilité de la convergence de l'algorithme selon le partage entre les deux types d'itérations; ce sujet sera traité dans la section suivante.

V.2 Discussion des critères d'arrêt et de partage entre les types d'itérations

Considérons notre problème sous sa forme générale:

(V.2.1) Min $f(x)$

sujet à

$$Ax = b$$

$$x \geq 0$$

où

$x \in E^n$, $f : x \to E^1$; $f \in C^1$ / $\{x \in Ax = b\}$.

L'expression de Taylor de f(x) peut s'écrire:

(V.2.2) $f(x+t\cdot d) = f(x) + t\cdot \nabla f(x)\cdot d + g(x,t\cdot d)\, \|t\cdot d\|$

où,

$t \in [0,1]$, $A\,(x+t\cdot d) = b$, $x+t\cdot d \geq 0$,

le terme $g(x,t.d)\, \|t.d.\|$

lorsque $t \to 0$.

On peut donc, à l'itération $\ell+1$, utiliser l'approximation:

(V.2.3) $f(x^{\ell+1}) = f(x^{\ell} + t^{\ell+1}(x-x^{\ell}))$

$\simeq f(x^{\ell}) + t^{\ell+1} . \nabla f(x^{\ell}) . (x-x^{\ell})$

Et, puisque l'on cherche à minimiser la valeur de la fonction économique, on a:

$\text{Min } f(x^{\ell+1}) = \text{Min } f(x^{\ell}) + \text{Min } t^{\ell+1}.\nabla f(x^{\ell}) . (x-x^{\ell})$

$= \text{const 1} + [t^{\ell+1} . \text{Min } \nabla f(x^{\ell}) . x - \text{const 2}]$

Ce qui revient à résoudre le sous-problème:

(V.2.4) $\text{Min } \nabla f(x^{\ell}) . x$

sujet à

$Ax = b$

$x \geq 0$

En notant $\bar{x}^{\ell+1}$ la solution de ce sous-problème, on peut établir

la correspondance entre ce dernier et le sous-problème (III.2.3). De plus $\nabla f(x^{\ell+1}) \cdot (\bar{x}^{\ell+1} - x^{\ell})$ doit être négatif pour que $(\bar{x}^{\ell+1} - x^{\ell})$ soit une direction de descente.

Considérons maintenant le problème consistant à trouver une borne inférieure à la valeur de la fonction économique. En fixant $t^i = 1$, on définit une plus grande borne inférieure:

$$\text{(V.2.5)} \quad \text{LWBND} = \max_{0 \le i \le \ell-1} \{f(x^i) + t^{i+1} \nabla f(x^i) \cdot (\bar{x}^{i+1} - x^i)\}$$

Le critère d'arrêt devient:

$$\text{(V.2.6)} \quad |(f(x^{\ell}) - \text{LWBND}) / f(x^{\ell})| \le \varepsilon .$$

Si cette condition n'est pas satisfaite, alors on détermine $x^{\ell+1} = x^{\ell} + t^{\ell+1} (\bar{x}^{\ell+1} - x^{\ell})$

où $t^{\ell+1}$ a été choisi de telle facon que $f(x^{\ell+1})$ est minimale. Ce critère est employé pour contrôler l'algorithme lors d'une itération régulière.

Par expérience on a constaté la relation:

$$\text{(V.2.7)} \quad |\nabla f(x^{\ell}) \cdot (\bar{x}^{\ell+1} - x^{\ell})| \ge |\nabla f(x^{\ell}) \cdot (y - x^{\ell})|$$

où

- $\bar{x}^{\ell+1}$ provient d'un "Toward step"
- y provient d'un "Away step".

Autrement dit, une direction basée sur un "Toward step" donne une meilleure amélioration de la valeur de la fonction économique qu'une direction basée sur un "Away step", en supposant $t^{\ell+1}$ identique pour les deux directions de descente.

C'est pourquoi on a choisi d'alterner entre ces deux directions de descente en se basant sur le numéro d'itération.

Maintenant nous discuterons du partage entre les itérations restrictives et les itérations régulières. Au début, on accumule L points extrêmes à partir d'itérations régulières puis l'on passe à des itérations restrictives.

Le critère de partage qui semble naturel se base sur l'amélioration potentielle de la valeur de la fonction économique (en fixant $t^{\ell} = 1$). On revient à une itération régulière lorsque:

$$|\nabla f(x^{\ell}) \cdot (\bar{x}^{j**} - x^{\ell}) / f(x^{\ell})| \leq \varepsilon \qquad \text{(V.2.8)}$$

où, $\bar{x}^{j**}$ est obtenu à partir d'un "Toward step". On préfère se baser sur un "Toward step" plutôt que sur un "Away step" à cause de la relation empirique (V.2.7) et aussi à cause du fait que l'on a constaté qu'un "Away step" est utile pour relancer la convergence aux itérations subséquentes.

Un autre critère peut se baser sur la comparaison de l'amélioration (relative à la solution courante) de la valeur de la fonction économique. Tant que cette amélioration est supérieure

pour un type d'itération à l'autre type, on fait des itérations d'un même type, puis on passe à l'autre type d'itération, en reprenant le même processus. Ce critère est utilisé dans le programme que l'on retrouve à l'annexe B.

V.3 Conclusion

En se basant sur les développements qui précèdent, on a obtenu des résultats pour la VERSION-2, supérieurs aux résultats de la VERSION-1 (originale), en terme de temps de calcul pour arriver à des solutions approximativement égales.

Ce nouvel algorithme profite du fait que les temps de calcul nécessaires aux itérations restrictives sont de beaucoup inférieurs aux temps de calcul impliqués dans l'évaluation des chemins les plus courts; de plus, les itérations restrictives aident à relancer la convergence des itérations régulières. En pratique on constate qu'il est avantageux de favoriser les itérations restrictives après un certain nombre d'itérations régulières.

Enfin, il est préférable de posséder à priori des informations et une vision intuitive du problème à résoudre par cette méthode. En choisissant les paramètres appropriés, on peut économiser avantageusement.

Dans la section suivante, nous améliorons le temps de calcul des sous-problèmes impliqués dans les itérations régulières.

CHAPITRE SIXIEME

AMELIORATION DU TEMPS DE CALCUL DE LA DIRECTION DE DESCENTE DE FRANK ET WOLFE

VI.1 Introduction

Il existe de nombreuses applications au calcul des chemins les plus courts d'un noeud du réseau à tous les autres noeuds du réseau. Les noeuds peuvent représenter les quartiers d'une ville, les intersections d'un réseau de rues, les points d'offre et de demande d'un système de distribution, etc...

Pour le problème en considération, la recherche d'une direction de descente revient à un calcul des chemins les plus courts ou les plus longs. La possibilité de rencontrer des cycles positifs dans le calcul des chemins les plus long, nécessite des calculs supplémentaires pour les retracer, et de l'espace-mémoire additionnel. C'est pourquoi on a utilisé la stratégie de restriction qui se limite à des lectures sur mémoires auxiliaires et au calcul de sommes et de produits.

En considérant le problème d'affectation de traffic pour le réseau de la ville de Winnipeg, on constate que pour 20 itérations de l'algorithme original de Frank et Wolfe, 68% du temps de calcul est utilisé par le calcul des chemins les plus courts; pour d'autres problèmes on observe aussi le même phénomène.

Ainsi, après avoir exploré des moyens d'améliorer la vitesse de convergence de l'algorithme de Frank et Wolfe, nous cherchons à diminuer le

temps de calcul à chaque itération régulière. Le choix de l'algorithme approprié pour le type de réseau sur lequel on opère, s'inspire de l'article écrit par Dial, Glover, Karney et Klingman [3] dans lequel ils comparent 9 algorithmes entre eux.

Après avoir introduit la terminologie nécessaire et motivé le choix de l'algorithme, nous le détaillerons et décrirons les résultats obtenus.

VI.2 Terminologie

Un graphe G=(N,A) consiste en un ensemble fini N de noeuds et un ensemble fini d'arcs A; on dénote chaque arc $a \in A$ par un couple ordonné (i,j), i étant le début de l'arc et j le noeud terminal (j est différent de i). Un chemin se définit par une séquence finie d'arcs $\{a_1, a_2, \ldots, a_n\}$ telle que pour tout $k = 2, \ldots, n$, l'arc a_k débute à la fin de l'arc a_{k-1}. Un chemin du noeud i au noeud j existe si l'arc a_1 débute au noeud i et l'arc a_n se termine au noeud j.

La fonction L(a) (ou encore L(i,j)) désigne la longueur non négative de l'arc a. Pour le sous-problème (III.2.3), $L(a) = s_a(v_a^{\ell})$. La longueur du chemin de i à j s'obtient par:

$$\sum_{k=1}^{m} L(a_k) \quad .$$

Une arborescence est un graphe $T=(N_T,A_T)$ avec une racine r tel que pour chaque noeud $x \in N_T$ (sauf r), il existe un chemin unique, partant de r et allant à x. Une arborescence pour laquelle tout chemin (entre r et x) est le chemin le plus court est une arborescence minimale du graphe

L=(N,A) (comprenant le graphe $T=(N_T,A_T)$).

VI.3 Représentation du réseau

Il existe trois représentations couramment utilisées. La représentation matricielle nécessite trop d'espace-mémoire ($|V| \times |V|$). La version originale du programme "TRAFFIC" (Nguyen et James [19]) contient une liste complète des arcs; noeud original, noeud terminal, longueur d'arc. L'espace-mémoire est raisonnable ($3|A|$) mais on retrouve plus facilement les successeurs d'un sommet avec la représentation qui suit.

Le troisième mode de représentation d'un réseau consiste à utiliser des pointeurs ("Forward star"). Les noeuds sont numérotés séquentiellement de 1 à $|N|$. Les arcs s'enregistrent consécutivement en mémoire de telle sorte que les arcs partant du noeud i suivent ceux qui partent du noeud i-1. Un emplacement de $|N|$ pointeurs et $2|A|$ arcs se traduisant par $|N| + 2|A|$ mots-mémoire suffit.

VI.4 Motivation du choix de l'algorithme

Les algorithmes de calcul des chemins les plus courts se classifient selon deux méthodes: méthode d'étiquetage basée sur la recherche de la prochaine étiquette permanente et méthode d'étiquetage basée sur la revision et la correction d'étiquette(s) temporaire(s).

La première méthode comprend l'algorithme de Dijkstra [4] et tous ses dérivés. A une itération quelconque, la prochaine étiquette permanente s'obtient en retenant l'arc dont le sommet terminal (étiqueté temporairement)

est le plus proche de la racine et dont l'extrémité initiale est étiquetée de façon permanente. Le point faible de cette méthode réside dans le fait qu'on doit effectuer un tri à chaque itération pour retrouver le sommet étiqueté temporairement le plus proche de la racine.

La seconde méthode comporte l'algorithme de Ford [4] et ses dérivées. On revise et corrige continuellement les étiquettes temporaires jusqu'à ce qu'il soit impossible d'effectuer d'autres changements; à ce moment, toutes les étiquettes deviennent permanentes. Toutes les améliorations proposées en vue d'éviter les revisions inutiles se rattachent à l'ordre dans lequel on doit reviser les sommets.

Les auteurs Dial, Glover, Karney et Klingman [3] étudient le comportement de 9 algorithmes (5 utilisant la première méthode, et 4 la seconde) sur deux types de réseaux:

(A) Un réseau sous forme de grillage où chaque noeud est en contact avec les quatre (s'ils existent) noeuds situés à gauche, à droite, au-dessus et en-dessous. Un grillage de dimension p x q contient 4.p.q-2.p-2.q arcs.

(B) Un réseau où l'on choisit aléatoirement les noeuds devant former une paire qui sera reliée par un arc.

Pour chaque type de réseau, ils ont fait varier la grandeur de l'intervalle des longueurs permises pour les arcs. Le modèle (A) se rapproche le plus du modèle de réseau routier d'une ville à cause de sa forme et de sa densité d'environ 4 arcs par noeud. On a retenu l'algorithme le

plus efficace pour ce type de réseau. Dès que l'on atteint des densités de l'ordre de 15 arcs par noeud, les algorithmes basés sur un tri efficace deviennent supérieurs à ceux de la lignée de Ford.

L'ancienne version du programme TRAFFIC contient un algorithme de Dijkstra avec un tri binaire. Dans la section suivante on présente l'algorithme de Dijkstra puis on discutera de l'algorithme de Ford et d'une version améliorée retenue pour le type de réseau que l'on traite.

VI.5 Algorithme de Dijkstra

Sans perte de généralité, on peut supposer que l'on cherche le plus court chemin du noeud 1 à tous les autres noeuds. Initialement, on associe au noeud 1 une étiquette permanente de valeur 0, et des étiquettes temporaires aux autres noeuds de valeur infinie (supérieure au plus long des plus courts chemins). Puis il s'agit, pour chaque noeud (différent de 1), de comparer la valeur de l'étiquette temporaire qui lui est associée avec la valeur de la distance directe du noeud 1 à ce dernier; on associe à chaque noeud une nouvelle étiquette temporaire étant le minimum de ces deux valeurs. Remarquons immédiatement que s'il n'y a pas de lien entre les sommets 1 et i, la valeur de la distance directe est infinie par convention.

Enfin, on déclare étiquette permanente la plus petite des $|N|-1$ étiquettes temporaires. Faisons l'hypothèse que l'étiquette associée au noeud k est devenue permanente. Alors on applique le processus qui suit: on compare la valeur de chacune des $|N|-2$ étiquettes temporaires avec la

somme de la valeur de l'étiquette associée au noeud k et la valeur de la distance directe entre le noeud k et le noeud en question. Pour chacun des |N|-2 noeuds, on associe une nouvelle étiquette temporaire étant le minimum de ces deux valeurs. On déclare étiquette permanente, la plus petite des |N|-2 étiquettes temporaires. On reprend le même processus à partir du sommet associé à cette nouvelle étiquette permanente.

La procédure se termine après |N|-1 applications du processus fondamental; toutes les étiquettes sont alors permanentes. On peut retracer les plus courts chemins entre les sommets 1 et r si on a retenu, à chaque fois que l'on donnait une étiquette permanente à un noeud, quel était le noeud précédent ayant une étiquette permanente. On peut retrouver dans l'article de Dreyfus [4] une preuve par induction de la validité de cette méthode.

Avant de formaliser l'algorithme, quelques hypothèses et définitions sont nécessaires:

- d(i) étiquette temporaire associée au noeud i.
- d*(i) étiquette permanente associée au noeud i.
- p(i) prédécesseur du noeud i; lorsque toutes les étiquettes sont permanentes on peut retracer les chemins les plus courts à l'aide de ce vecteur.
- L(i,j) longueur de l'arc dont l'extrémité initiale est i et l'extrémité terminale est j; $L(i,j) = \infty$ s'il n'existe pas d'arc entre i et j; $L(i,j) \geq 0$.

- N* l'ensemble des sommets i tels que l'étiquette associée au noeud i soit permanente.
- N l'ensemble des sommets du graphe G=(N,A).
- |N| dénote le nombre d'éléments dans l'ensemble N.

On n'a pas nécessairement $L(i,k) + L(k,j) \geq L(i,j)$ et $L(i,j) = L(j,i)$.

Voici l'algorithme de Dijkstra qui trouve les plus courts chemins entre le noeud 1 et tous les autres noeuds, tout en permettant d'avoir l'information nécessaire pour les retracer:

Etape 0: $N^* = \{1\}$; $i^* = 1$; $d^*(i^*) = 0$;
$d(j) = \infty$, $j=2,\ldots, |N|$;
$p(j) = 0$, $j=1,2,\ldots|N|$;

Etape 1: pour tout n tel que $n \notin N^*$ et $n \in N$,
$d(n) = \{\text{Min } d(n), d^*(i^*) + L(i^*,n)\}$;

Etape 2: trouver n* tel que

$$d(n^*) = \underset{\substack{n \in N \\ \text{et} \\ n \notin N^*}}{\text{Min}} \{d(n)\}$$

Etape 3: $N^* = N^* \cup \{n^*\}$;
$d^*(n^*) = d(n^*)$;
$p(n^*) = i^*$;

Etape 4: si $N^* \neq N$ alors
$i^* = n^*$, aller à l'étape 1
sinon, on a terminé.

Pour retracer le plus court chemin du noeud 1 au noeud i, il s'agit de remonter en partant de p(i) jusqu'à p(1) = 0.

Si on cherche à estimer le nombre d'opérations, alors on compte ($|N|$. ($|N|$-1)/2) = $\sum_{i=1}^{|N|-1} i$ additions et comparaisons pour rendre les étiquettes permanentes (cf. étape 1). De plus, on doit ajouter ($|N|$. ($|N|$-1)/2 comparaisons pour trouver tous les n* de l'étape 2. Donc, de l'ordre de $|N|^2/2$ additions et $|N|^2$ comparaisons.

Il existe une autre preuve de la validité de l'algorithme donnée par HU [13]. De même on peut adapter un théorème démontré par Potts-Oliver [21].

> "les chemins les plus courts du noeud 1 à tous les autres noeuds r d'un graphe G=(N,A) avec des longueurs d'arcs positives L(i,j) sont l'unique solution des équations
>
> $$d(1) = 0 ;$$
>
> $$d(r) = \min_{\substack{i \neq r \\ r \neq 1}} \{d(i) + L(i,r)\} .$$
>
> "

VI.6 Amélioration de l'algorithme de Ford

En utilisant la notation définie dans la section VI.5, l'algorithme de Ford peut s'écrire:

Etape 0: $N^* = \{1\}$; $d(1) = 0$:

$d(j) = \infty$, $j=2,\ldots,|N|$

$p(j) = 0$, $j=1,2,\ldots,|N|$

Etape 1: trouver i,j tel que

$$d(i) + L(i,j) < d(j) ;$$

Etape 2: si i et j existent alors

$$d(j) = d(i) + L(i,j)$$

$$p(j) = i$$

retourner à l'étape 1

sinon

(* on a terminé, toutes les étiquettes deviennent permanentes*)

$$d^*(i) = d(i) \qquad i=1,2,\ldots,|N| :$$

On évite des comparaisons et additions aux étapes 1 et 2 en modifiant la structure de l'algorithme; en effet:

(1) on effectue des opérations inutiles lorsque l'on recherche i et j à l'étape 1 et qu'on a $d(j) = \infty$,

(2) si pour un noeud quelconque i, on a la relation

$$d(i) + L(i,j) \geq d(j)$$

pour tout j tel que (i,j) est un arc, alors il est inutile de considérer ces arcs tant que d(i) ne diminue pas.

Donc, il devient intéressant de garder une liste des noeuds pour lesquels la fonction d(j) doit diminuer, sachant que d(i) diminue et qu'il existe un arc entre i et j.

La gestion de la liste consiste à choisir selon le mode FIFO ("First in, first out") les noeuds au début de la liste, et à ajouter les nouveaux

noeuds à la fin de la liste. On efface toujours le noeud au début de la liste s'assurant ainsi d'au plus une apparition pour chaque noeud.

L'implantation de cette liste séquentielle est conforme à l'idée de Pape [20]. On utilise un tableau noté CL(i), indexé par un numéro correspondant à un noeud. La convention utilisée est:

$$CL(i) = \begin{cases} \text{. } -1 \text{ si le noeud i a déjà fait partie de la liste mais n'y apparaît plus.} \\ \text{. } 0 \text{ si le noeud i n'a jamais fait partie de la liste.} \\ \text{. } +j \text{ si le noeud i est dans la liste et le noeud j est son successeur.} \\ \text{. } \infty \text{ si le noeud i est le noeud terminal de la liste.} \end{cases}$$

Il est aussi possible de contrôler les noeuds devant faire partie de la liste (par exemple, lorsqu'on interdit le passage des chemins les plus courts par des centroïdes). De plus, le troisième mode de représentation discuté dans la section VI.3 s'avère le plus approprié à cause du besoin d'accéder aux successeurs de chaque noeud.

La supériorité d'un algorithme basé sur un tel traitement des sommets à considérer, s'explique par le fait que les opérations inutiles décrites en (1) et (2) sont évitées et aussi par la structure même du réseau.

En effet, toute arborescence d'un réseau sous forme de grille (ayant une densité faible d'arc par sommet) est étroite et allongée, ainsi que toute sous-arborescence. Donc, si pour un noeud i, d(i) décroît, alors tous les noeuds j qui en découlent, sont affectés, et il devient

profitable de mettre à jour les valeurs des d(j) le plus tôt possible, avant même de considérer d'autres noeuds. C'est exactement ce que fait l'algorithme en plaçant le noeud i au début de la liste.

Avant de présenter la version améliorée de l'algorithme de Ford nous allons illustrer son fonctionnement. Supposons que l'on possède une liste séquentielle de sommets à considérer. Les développements qui précèdent conduisent à la procédure:

Etape 0: [initialisations]

. d(r) = 0;

. d(i) = ∞ pour tout i, i ≠ r

. i = r

(* i est un noeud dont le potentiel aura à diminuer étant donné que le potentiel de l'un de ses prédécesseurs a diminué*)

. i est le seul noeud dans la liste séquentielle des candidats;

Etape 1: pour chaque j ∈ {successeurs de x} faire

si d(i) + L(i,j) < d(j) alors début

(I) on décroît le potentiel du noeud j: d(j) = d(i) + L(i,j)

(II) si le noeud j n'a jamais fait partie de la liste des candidats alors on l'ajoute à la fin de la liste.

(III) si le noeud j a déjà fait partie de la liste des candidats, alors on l'ajoute au début de la liste.

(IV) si le noeud j fait partie de la liste des candidats alors on le laisse en place.

fin;

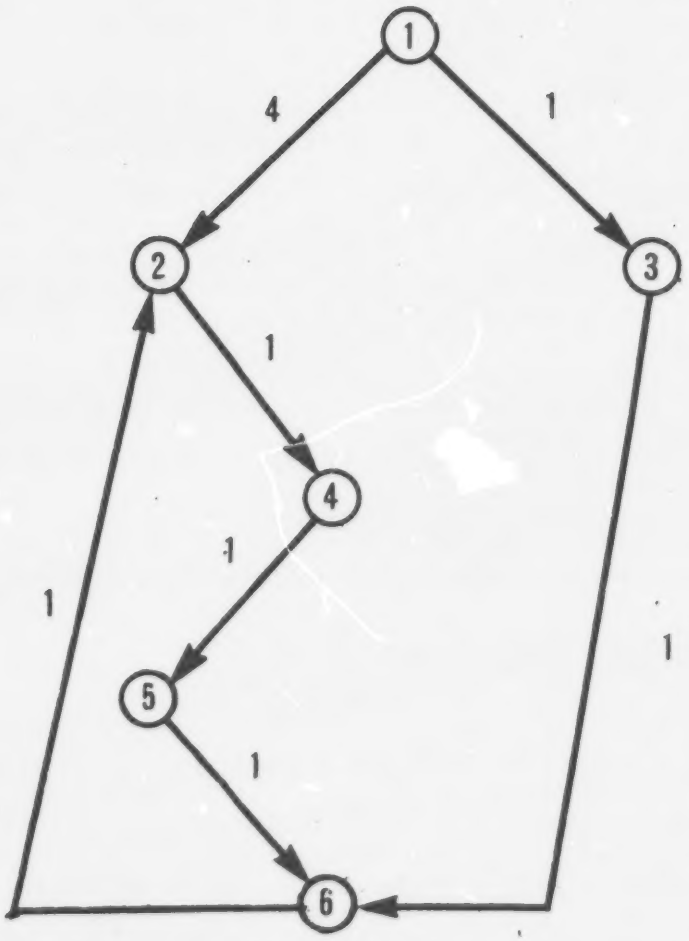


FIGURE [1]

Etape 2: - tous les successeurs du noeud i sont dans la liste des candidats car leur potentiel a diminué;
- on retient le fait que le noeud i a déjà fait partie de la liste des candidats;
- on redéfinit i comme étant le premier élément de la liste des candidats;
- si la liste est non vide alors on retourne à l'étape 1.

Considérons la figure 1. On cherche à déterminer le chemin le plus court du noeud 1 à tous les autres noeuds. Ce graphe comporte un cycle.

Si on enregistre dans une liste les noeuds candidats selon leur ordre d'occurrence, on considère d'abord le noeud 1, puis ses successeurs soit les noeuds 2 et 3. Ensuite on met à jour des étiquettes (d(i)) des noeuds 4 et 6. Suivent les mises à jour pour les noeuds 5 et 2. Enfin on corrige les étiquettes des noeuds 4 et 5. Ainsi on modifie deux fois les étiquettes associées aux noeuds 4 et 5.

D'autre part, si on utilise la gestion de la liste des candidats telle que décrite dans la procédure précédente, on donne priorité par la règle III de l'étape 1 aux anciens noeuds. Les modifications d'étiquettes associées aux noeuds s'effectuent dans l'ordre suivant: 1, 2, 3, 4, 6, 2, 4, 5. Ainsi, dès que le noeud 2 devient de nouveau candidat, on le place au début de la liste, évitant de faire une mise à jour de l'étiquette du noeud 5 inutilement. Donc, la règle III est une politique ayant pour but d'éviter des corrections d'étiquettes et qui s'applique lorsqu'on a des cycles. De plus, la règle II nous dit d'avancer progres-

sivement dans le réseau. Cette règle est particulièrement efficace dans le cas où les arcs sont à peu près de même longueur. En effet, on évite de s'aventurer trop profondément dans le réseau par un chemin quelconque pour être ensuite obligé de corriger de nouveau les étiquettes s'il existait à l'origine un autre chemin plus court. La règle IV respecte l'ordre établi par les règles précédentes.

Si on utilise l'algorithme de Dijkstra pour résoudre le problème de la figure 1, alors on modifie les étiquettes associées aux noeuds dans l'ordre suivant: 1, 2, 3, 6, 2, 4, 5. On avance progressivement en fonction de la longueur des chemins. On évite des corrections d'étiquettes mais on doit comparer les longueurs des chemins entre elles.

La version améliorée reprend la procédure décrite plus haut en précisant la nature de la liste séquentielle. Cette dernière est symbolisée par un tableau CL dont les éléments CL(i) sont définis tel que décrit précédemment. De plus cette version utilise le "Forward star notation"

Version améliorée

Etape 0: [initialisations; NN est la racine]

(.) D(i) = ∞, Npr(i), CL(i) = 0 i=1,..., nombre de noeuds

(.) D(NN) = 0;

(.) CL(NN) = ∞;

(* NN est le noeud terminal de la liste*)

(.) i = NN;

(* i représentera un noeud dont le potentiel d(i) aura diminué*).

```
(.) NT = NN;
            (* NT est un pointeur sur le noeud terminal de la
               liste*)

Etape 1: [boucle principale: considère tous les successeurs du sommet i]
pour ir = NOLAB(i),..., (NOLAB(i+1) - 1
(.) K = ND(ir);
            (* on examine tous les arcs (i,k) dont le sommet
               initial est i*).
(.) si D(i) + trav(ir) < D(K) alors
            (* on peut décroître le potentiel du noeud K en
               passant par l'arc (i,K)*)
   (..) D(K) = D(i) + trav (ir);
   (..) Npr = i;
            (* le noeud précédent le sommet K est le noeud i*)
   (..) Lpr(K) = ir;
            (* le noeud terminal de l'arc (i,K) est K= ND(ir);
               la longueur de l'arc (i,K) est trav (ir)*)
   (..) cas  CL(i)  de
            (* vérifie le statut du noeud K dans la liste*)
     [1] CL(K) = 0
         si K ≥ NCENT alors
            (* un chemin ne doit pas passer par un centroïde;
               le noeud K est ajouté à la fin de la liste et
               devant le nouveau noeud NT*)
         (*) CL(NT) = K,
```

(*) CL(K) = ∞

(*) NT = K

[2] CL(K) < 0

(* le noeud K a déjà été visité; on l'ajoute en tête de liste*)

(*) CL(K) = CL(i),

(*) CL(i) = K

[3] CL(K) > 0

(* le noeud K fait déjà partie de la liste*)

Etape 2: [choisit le prochain sommet i dont le potentiel D(i) a diminué]

(.) ICL = CL(i)

(.) CL(i) = -1

(*le noeud i a déjà été considéré*)

(.) i = ICL

(* i devient le noeud de tête de liste*)

(.) si (i<∞) alors

retourner à l'étape 1

sinon

(* il ne reste plus de noeud à considérer*)

on a terminé.

VI.7 Comparaison des performances

Le programme intitulé "Traffic" (Nguyen et James [19]) utilise une sous-routine pour le calcul des chemins les plus courts basée sur l'algorith-

me de Dijkstra. Cependant on y retrouve une structure d'ordre binaire ayant pour racine la variable n*, dont la valeur est déterminée à l'étape 2 de l'algorithme de Dijkstra.

Cette sous-routine se retrouve sous deux versions différentes selon le second mode de représentation (version originale TRAFFIC) ou selon le troisième mode de représentation (version TRAFFIC II).

Les résultats de la comparaison apparaissent au tableau II et ont été obtenus àl'aide d'un ordinateur CDC 173. On a dû se limiter à une seule itération pour le réseau de Winnipeg à cause du fait que pour un couple origine-destination, il existait au départ deux plus courts chemins passant par des arcs différents. Les sous-routines n'utilisant pas le même, il s'en suivait que le volume de déplacements sur les arcs varient selon la sous-routine utilisée; le coût associé à un arc est fonction du volume de déplacements sur l'arc en question, par conséquent, à la seconde itération, le calcul des chemins des plus courts s'effectue sur un réseau différent selon la sousroutine utilisée précédemment; il devient donc impossible de faire une comparaison sur plus d'une itération. Ce problème ne se pose pas pour les autres réseaux. On constate des améliorations de l'ordre de 53% à 63.7%. Les résultats confirment, pour le type de réseau que l'on utilise, les conclusions de l'article [3]. Les sousroutines se retrouvent en annexe.

A cause de la différence de représentation on sauve environ 2096 mots mémoire (5204 pour la version améliorée, 7300 pour la version "TRAFFIC"), pour le réseau de Winnipeg. Cependant, si on compare les versions uti-

Tableau II

Comparaison de deux algorithmes de calcul des chemins les plus courts sur trois réseaux.

Réseau	Steenbrink	Hull	Winnipeg
Compilateur	FTN(opt=1)	FTN(opt=1)	FTN(opt=2)
Nombre d'appels	22	34	1
Dijkstra structure d'arbre binaire (secondes)	.15	35.39	35.12
Fond, Version améliorée (secondes)	.07	12.85	15.89
% d'amélioration	53.34	63.70	54.76

lisant le troisième mode de représentation du réseau, on s'aperçoit qu'approximativement le même espace mémoire est nécessaire; la différence réside au niveau de l'enregistrement des noeuds dont le potentiel D(i) a diminué: la version améliorée comporte un tableau CL de dimension égale au nombre de noeuds dans le graphe tandis que l'ancienne version nécessite deux tableaux de dimension arbitraire (l'espace mémoire total fixé dans les cas pratiques s'élève à 20% du nombre de noeuds). Pour l'algorithme de Frank et Wolfe (version originale ou améliorée) appliqué à un graphe (G(N,A), environ 5.|N|+2.|A| mots mémoire suffisent pour employer la version améliorée de la sous-routine des chemins les plus courts.

CHAPITRE SEPTIEME

CONCLUSION

Il existe d'autres alternatives pour résoudre un problème non linéaire. On a développé l'approximation linéaire de la fonction économique telle que suggérée par Frank et Wolfe, combinée avec l'utilisation de deux directions de descente ("Away" et "toward" steps avec restriction).

Cantor et Gerla [22] ont effectué des essais avec la méthode du gradient projeté. Pour des contraintes linéaires, étant donné une solution réalisable, cette méthode primale consiste à projeter sur une variété linéaire (frontière) du domaine, le gradient de la fonction économique évalué en ce point et à se déplacer dans la direction inverse du gradient. Pour des déplacements à l'intérieur du domaine, on fait appel aux méthodes de gradient usuelles. Cette méthode est avantageuse lorsque la solution optimale se trouve sur une des frontières du domaine.

On peut aussi utiliser la méthode du gradient réduit que l'on retrouve dans Luenberger [16]. Nos contraintes sont linéaires; on peut aisément retracer pour une solution en main, les variables de base (dépendantes) et les variables hors-base (indépendantes). L'idée de cette méthode se résume ainsi: on considère à chaque itération le problème par rapport aux variables indépendantes. La fonction économique peut s'exprimer en terme des variables indépendantes uniquement. On évalue son gradient (d'où le nom 'gradient réduit' aux variables indépendantes). On se déplace dans la direction inverse de ce gradient. Si au cours de ce déplacement une variable indépendante s'annule alors on doit calculer une

autre direction. D'autre part, si au cours de ce déplacement une variable dépendante s'annule, alors on doit modifier la définition des variables de base et hors-base: la variable dépendante qui s'annule devient hors-base (indépendante) et une variable indépendante à valeur positive devient de base (dépendante). On retrouve une adaptation de cette méthode dans l'article: "An improved linear approximation algorithm for the network equilibrium (packet switching) problem" par Florian [6].

Un des objectifs de cette thèse consistait à améliorer le temps d'exécution de l'algorithme de Frank et Wolfe. Puisqu'une grande part des calculs se retrouvaient dans la recherche d'une nouvelle direction de descente, il était nécessaire de reconsidérer l'algorithme de calcul des chemins les plus courts. De fait, pour le genre de réseau que nous utilisons, une procédure basée sur la correction d'étiquettes temporaires s'avérait plus efficace. Les résultats pratiques démontrent, pour des essais sur des réseaux ayant une faible densité d'arc par sommet, des améliorations d'au moins 53% du temps de calcul.

Un autre objectif portait sur l'amélioration de la convergence de l'algorithme de Frank et Wolfe. La démarche suivie a été l'extension de la théorie sur la méthode des directions réalisables, sur la stratégie de restriction et sur la linéarisation interne du domaine, en vue de développer deux nouvelles directions de descentes (moins coûteuses): "Away step restrictif" et "Toward step restrictif".

La méthode de Frank et Wolfe possède pour les premières itérations, un taux de convergence rapide, puis se stabilise donnant un taux lent. En

tenant compte de ce fait et de la nécessité d'accumuler un "certain" nombre de points de base pour la stratégie de restriction, il faut effectuer initialement un nombre fixe d'itérations régulières, selon l'algorithme de Frank et Wolfe. En effet, il faut choisir une stratégie consistant à répartir les itérations régulières (chacune d'elle ajoutant un nouveau point de base) et les itérations restrictives. Le critère de choix entre les types d'itérations se base sur la diminution potentielle de la valeur de la fonction économique.

Il est difficile de comparer l'algorithme de Frank et Wolfe et l'algorithme amélioré. En effet, le premier est une procédure automatique tandis que le second fait appel à une stratégie et peut varier dans ses performances selon l'intuition de l'usager. Les résultats pratiques démontrent que l'utilisation d'une stratégie "raisonnable" améliore la convergence de l'algorithme de Frank et Wolfe en ce qui a trait au nombre d'itérations et au temps de calcul pour arriver à une solution semblable.

En plus d'atteindre les objectifs mentionnés plus haut, ce rapport de stage établit de façon détaillée le lien entre le modèle de réseau et le modèle général du problème de minimisation d'une fonction économique non linéaire, sous des contraintes linéaires; on y retrouve une extension et une application de la théorie se rapportant à la stratégie de restriction et à la linéarisation interne du domaine formé par des contraintes linéaires.

ANNEXE A

Exemple de formulation

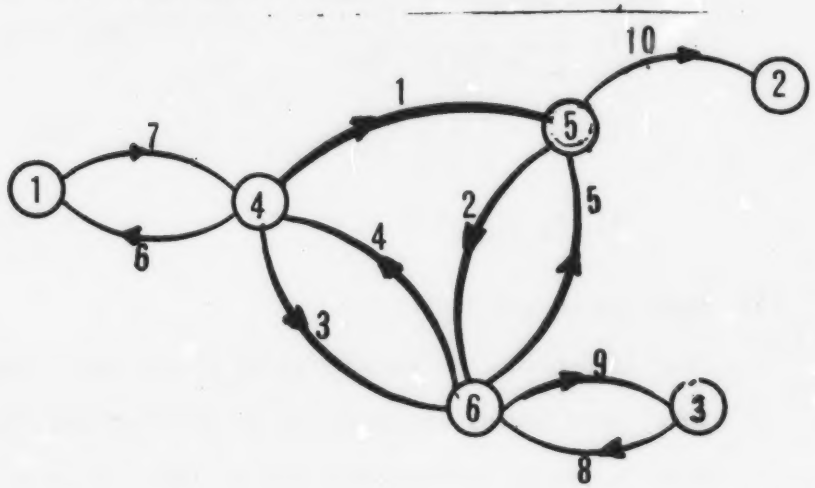
EXEMPLE DE FORMULATION (Potts-Oliver [14])

(1) Réseau:

Ce réseau est représenté sous la forme d'un graphe G(N,A) où

. N = {1,...,6} est l'ensemble des noeuds; les trois premiers sont des centroïdes

. A = {1,...,10} est l'ensemble des arcs.



(2) Matrice origine-destination

La matrice origine-destination $[g_{pq}]$, $p \in A = \{1,2,3\}$, $q \in Q = \{1,2,3,\}$ associée à ce réseau est:

p \ q	1	2	3
1	0	3	6
2	0	0	0
3	2	5	0

(3) Equations générales:

Les équations générales selon la représentation arc-chemin de ce problème d'affectation de trafic à plusieurs produits (origines, destinations) à coûts convexes sont:

(1) Min $\sum_a \int_0^{v_a} s_a(v)\, dv = (\sum_s S_a(v_a))$

sujet à

(2) $\sum_k h_{k,(pq)} = g_{pq}$ $\quad p\in P,\ q\in P,\ k\in K,(pq)$

(3) $v_a = \sum_p \sum_q \sum_k \delta_{(ak),(pq)} \cdot h_{k,(pq)}$ $\quad a\in A$

(4) $h_{k,(pq)} \geq 0$ $\quad k,(pq) \in K,(pq)$

(4) Description des équations:

Pour le problème en considération il y a sept chemins possibles au total pour toutes les origines et destinations; notons par des couples $(k,(pq))$, les éléments des ensembles $K,(pq)$; on a:

- $K,(12) = \{(1,(12)),\ (2,(12))\}$
- $K,(12) = \{(1,(13)),\ (2,(13))\}$
- $K,(31) = \{(1,(31))\}$
- $K,(32) = \{(1,(32)),\ (2,(32))\}$
- $K,(pq) = \{\emptyset\}$ pour les autres (pq)

De façon exhaustive:

k	p	q	arcs faisant partie du chemin k,(pq)			
1	1	2	7	1	10	
2	1	2	7	2	5	10
1	1	3	7	1	2	9
2	1	3	7	3	9	
1	3	1	8	4	6	
1	3	2	8	5	10	
2	3	2	8	4	1	10

Les variables décisionnelles $h_{k,(pq)}$ doivent prendre des valeurs telles que la quantité cumulative de déplacements alloués sur tous les chemins de l'origine p vers la destination q respecte la demande en déplacements g_{pq} entre l'origine p et la destination q, et ce, pour tout couple (pq), comme l'indique l'équation (2).

Les variables v_a représentent le nombre total de déplacements sur l'arc a. La valeur d'une variable v_a s'obtient en additionnant le nombre de déplacements passant par les chemins qui utilisent l'arc a. Pour ce faire, on définit une fonction indicatrice notée $\delta_{ak,(pq)}$ qui a pour valeur 1 si l'arc a appartient au $k^{ième}$ chemin associé au couple origine-destination (pq), et pour valeur 0 sinon. Par exemple, pour l'arc numéro 1:

$$\delta_{1k,(pq)} = \begin{cases} 1 \text{ pour les couples } .(1,(21)) \\ \qquad\qquad\qquad\quad .(1,(31)) \\ \qquad\qquad\qquad\quad .(3,(22)) \\ 0 \text{ pour les autres.} \end{cases}$$

Les regroupements décrits par l'équation (3) sont nécessaires à cause de la définition des coûts de transport.

L'objectif consiste à trouver une répartition optimale des déplacements du point de vue des usagers. Les fonctions $s_a(v_a)$ sont convexes et non décroissantes. Par exemple

$$s_a(v_a) = \begin{cases} 1/(5-v_a) & \text{pour } a = 1,2,\ldots,5 \\ & 0 \le v \le 5 \\ 0 & \text{pour } a = 6,7,\ldots,10 \end{cases}$$

$$S_a(v_a) = \begin{cases} \int_0^{v_a} s_a(v)\, dv = (1/v_a) \ln (5/(5-v_a)) & \text{pour } a = 1,2,\ldots,5 \\ 0 & \text{pour } a = 6,7,\ldots,10 \end{cases}$$

ANNEXE B

Programme utilisé pour le réseau de la ville de Winnipeg

```
      PROGRAM TRAFFIC    73/173  OPT=1                    FTN 4.6+460        21/06/73

      PROGRAM TRAFFIC (INPUT,OUTPUT,TAPE6=INPUT,TAPE1,TAPE8=640,
     +TAPE7=640,
     +       TAPE9=640)
C
C--------THIS PROGRAM PERFORMS THE ASSIGNMENT OF TRAFFIC TO LINKS IN A
C        GIVEN TRANSPORTATION NETWORK TO SATISFY THE DEMAND BETWEEN THE
C        ORIGIN/DESTINATION PAIRS IN THE NETWORK SUCH THAT THE COST TO
C        EACH INDIVIDUAL USER F THE NETWORK IS MINIMIZED.
C
C--------THE PROGRAM IS BASED ON THE ALGORITHM DESCRIBED IN
C        "A MATHEMATICAL PROGRAMMING APPROACH TO EQUILIBRIUM METHODS OF
C        TRAFFIC ASSIGNMENT WITH FIXED DEMANDS" BY SANG NGUYEN
C        (UNIVERSITE DE MONTREAL,DEPT DE INFORMATIQUE,
C        TECHNICAL REPORT, NUMBER 138, AUGUST 1973).
C
      COMMON /NET1/ NNODE,NLINK,NCENT,NACTIV,KD(150),NOLAB(1300),
     +              ND(3000),NULACO(3000),DIST(3000)
      COMMON /NET2/ VCUM(3000),TRAV(3000), FLOW(3000),ODIJ(150)
      COMMON /MIN/ LPR(1301),NPR(1301)
      COMMON /PARAM/ EPSI , TIMEMAX, MAXCYC, MAXIT, IPCH, IPRT,IPOST
      COMMON /STAT/ RUNSEC(10), ITERAT(10)
      COMMON /COEF/ ALPHA(20),BETA(20),GAMMA(20),DELTA(20)
      COMMON /EDIT/ LINE(11),L(26),K(26),TITLE(10)
      COMMON /UNIT/ CARD,ODFIL,NETFIL,SOLFIL
      INTEGER CARD,ODFIL,NETFIL,SOLFIL
      REAL LAMBDA,ISLSEC
      EQUIVALENCE (NORIG,LINE(1)),(NDEST,LINE(2)),(IDIST,LINE(3)),
     +            (NULA,LINE(4)),(DEFUNC,LINE(5)),(ISPEED,LINE(6)),
     +            (ITOBS,LINE(7)),(IVOBS,LINE(8)),(LTYPE,LINE(9))
      EQUIVALENCE (RDSEC,RUNSEC(1)),(ISLSEC,RUNSEC(2)),
     +            (PRTSEC,RUNSEC(3)),(GSMSEC,RUNSEC(4)),
     +            (PTHSEC,RUNSEC(5))
C
      DATA CARD,ODFIL,NETFIL,SOLFIL/5,1,8,7/
      DATA MXRUNS/5/
C
      DATA ALPHA/.07730,.03350,.02960,.01970,.01030,.03250,.02910,
     +.01920,.00994,.02690,0.,0.,0.,.01810,.00943,.02350,.01670,.00863,
     +.00863,0/
      DATA BETA/39.973129,26.918268,23.768789,13.270287,8.077084,
     +25.651360,21.099385,15.050774,4.697130,16.497108,0.,0.,0.,10.030717
     +,2.637957,9.792157,10.786838,2.203220,2.0690,0/
      DATA GAMMA/280.0505,504.6483,727.8269,984.82,1222.4357,534.2537,
     +788.8415,1054.6928,1254.0953,804.8975,0.,0.,0.,1079.2004,1258.5142,
     +810.4795,1124.7455,1287.1012,1666.7,0/
      DATA DELTA/1.339569,1.863010,2.054611,2.393925,2.207982,1.274428,
     +1.595482,1.682623,1.777982,1.232178,0.,0.,0.,1.394251,1.398494,
     +0.739413,0.784880,0.933097,1.107,0/
C
 1002 FORMAT(*0101 RUN TERMINATED.....TIME LIMIT :*,E15.7,//)
 1003 FORMAT(*0102 RUN TERMINATED ... CYCLE LIMIT :*,I8,/)
 1013 FORMAT (*0 EPSILON =*,E15.7/*  TIMEMAX =*,E15.7/
     +        *  MAXCYC  =*,I8)
 1014 FORMAT (*0TOTAL VEHICLE HOURS =*,F12.2/
     +        *0TOTAL VEHICLE MILES =*,F12.2/
     +        *0AVERAGE SPEED       =*,F12.2//)
 1019 FORMAT (*0 CPU SECS FOR - NETWORK DATA INPUT*,T48,*=*,F10.2/
```

```
      PROGRAM TRAFFIC    73/173  OPT=1                          FTN 4.6+460        21/0
     +        T16,*- SET UP INITIAL SOLUTION*,T48,*=*,F10.2/
     +        T16,*- RE-ESTABLISH PARTIAL SOLUTION*,T48,*=*,F10.2/
     +        T16,*- GOLDEN SECTION METHOD (TOTAL)*,T48,*=*,F10.2/
     +        T16,*- FINDING SHORTEST PATH (TOTAL)*,T48,*=*,F10.2//)
 2000 FORMAT (1H1,100X,*PAGE :*,I4//30X,4A10/18X,6A10//
     +        4X,*LINK     NODE     NODE     LENGTH     NUMB    DELAY*,
     +        4X,*OBSERVED    OBSERVED    OBSERVED     LINK*,
     +        2X,*PREDICTED  PREDICTED*/
     +        4X,*IDEN     FROM      TO*,14X,*LANES   FUNCT     TRAVEL*,
     +        6X,*TRAVEL      VOLUME      TYPE   TRAVEL     VOLUME*/
     +        55X,*SPEED*,8X,*TIME*,26X,*TIME*/
     +        29X,*MILES*,21X,*(MPH)*,7X,*(MINS)*,24X,*(MINS)*//)
 2003 FORMAT (1H1,30X,4A10/18X,6A10//)
 2005 FORMAT (4X,3(I4,4X),F6.2,4X,F3.1,5X,I2,8X,F4.1,6X,F6.2,4X,I8,
     +        7X,I2,3X,F6.2,7X,I5)
 2004 FORMAT(5X,3(I4,1X),I3,I2,1X,I4,1X,2(I5,1X),3(I3,1X))
 2006 FORMAT (4X,3(I4,4X),F6.2,4X,F3.1,5X,I2,8X,F4.1,6X,F6.2,4X,A8,
     +        7X,I2,3X,F6.2,7X,I5)
 2011 FORMAT(10F8.0)
 2012 FORMAT (10F8.2)
 2013 FORMAT (20I4)
 3000 FORMAT (*0108 END CYCLE *,I4,* OBJ FUNCTION =*,E16.10,
     +        * RATE OF CHANGE =*,E16.10,* CUMULATIVE RUN TIME =*,F8.2)
 3003 FORMAT(*1109 INITIAL SOLUTION-VALUE OF OBJ. FUNCTION*,E20.10)
 3004 FORMAT(2X,*I10 DEL(F) < EPSILON *,E20.10)
 4001 FORMAT(10A8)
*
*  INITIALISATION
*
      CALL SECOND (TBEGIN)
      DO 50 I=1,10
      RUNSEC(I)=0.
   50 ITERAT(I)=0
*--------READ NETWORK DATA
      CALL NETWORK
      CALL SECOND(TTOUT)
      RDSEC=TTOUT-TBEGIN
      TTIN =TTOUT
*--------CHECK FOR POST OPTIMIZATION
      IF(IPOST.LT.1)GOTO 60
          CALL PARTSOL
          CALL SECOND(TTOUT)
          PRTSEC = TTOUT - TTIN
          GO TO 70
*
*--------SET UP INITIAL (ALL OR NOTHING) SOLUTION
*
   60 REWIND ODFIL
      DO 40 I=1,NLINK
          VCUM(I)=0.0
   40     TRAV(I)=USERCST(I)
      CALL EXFLOW
      DO 43 I=1,NLINK
          VCUM(I)=FLOW(I)
   43 CONTINUE
      CALL SECOND(TTOUT)
      ISLSEC=TTOUT-TTIN
```

```
      PROGRAM TRAFFIC    73/173  OPT=1                          FTN 4.6+460        21/06/
115           *
                 70 CONTINUE
                    TTIN=TTOUT
                    KOPT=0
                    NUMCYC=1
120                 IF(NUMCYC.EQ.1) GOTO 300
              *
              *--------CALCULATE OBJECTIVE FUNCTION FOR CURRENT SOLUTION
              *
                    DO 71 I=1,19
125              71     FLOW(I)=0.0
                    DO 72 I=1,19
                    IF(I.GT.10.AND.I.LE.13)GOTO 72
                        AG=ALPHA(I)*GAMMA(I)
                        AG2=AG*AG
130                     RAC=SORT(AG2+BETA(I))
                        VAL=ALOG( RAC-AG)
                        FLOW(I)=0.5*(GAMMA(I)*RAC - BETA(I)/ALPHA(I)*VAL)
                 72 CONTINUE
                    CONST=0.0
135                 DO 73 J=1,NLINK
                        I=MOD(NULACO(J),1000)
                        IF(I.GT.10.AND.I.LE.13)GOTO 73
                            NULA=NULACO(J)/1000
                            AL=NULA/10.0
140                         VAL=AL*(FLOW(I)-0.5*BETA(I)/ALPHA(I)*ALOG(AL))
                            CONST=CONST + 0.01*DIST(J)*VAL
                 73 CONTINUE
                    OBF=SUMTRAV (0.00) + CONST
              *--------CONST = CONSTANT TERM OF OBJECTIVE FUNCTION
145           *              = TOTAL COST, GIVEN USERCST VALUES CALCULATED FOR ZERO
              *                VOLUMES
              *--------SUMTRAV(0.0) = VARIABLE TERM OF OBJECTIVE FUNCTION
              *                     = TOTAL COST, GIVEN USERCST VALUES CALCULATED
              *                       USING INITIAL SOLUTION VOLUMES.
150           *                                        (LAMBDA = 0.0)
                    PRINT 3003,OBF
              *
                200  CONTINUE
              *
155           *--------ACCUMULATE TRIAL FLOWS FOR NEW SOLUTION USING CURRENT
              *         SOLUTION VOLUMES TO CALCULATE COST
              *
                    CALL EXFLOW
              *
160           *--------CALCULATE INSTANTANEOUS RATE OF CHANGE OF OBJECTIVE FUNCTION
              *         ALONG THE DESCENT DIRECTION
              *
                    SUM=0.00
                    DO 202 I=1,NLINK
165                     FLOW(I)=FLOW(I)- VCUM(I)
                202     SUM=SUM+ TRAV(I)*FLOW(I)
              *
              *--------FLOW(I) IS NOW DESCENT DIRECTION FOR LINK I
              *--------FIND LAMBDA TO MINIMIZE OBJECTIVE FUNCTION ALONG THE CURRENT
170           *         DESCENT DIRECTION.
              *         VAL + CONST = VALUE OF THE OBJECTIVE FUNCTION FOR THE
```

```
      *                    OPTIMUM LAMBDA.
            ACCUR=0.0001
            CALL SECOND(TTIN)
            CALL GSM(LAMBDA,1.00,ACCUR,VAL)
            CALL SECOND(TTOUT)
            GSMSEC = GSMSEC +TTOUT-TTIN
      *
      *  REVISE THE CURRENT LINK FLOW
            DO 203 I= 1,NLINK
            VCUM(I)=VCUM(I) + LAMBDA*FLOW(I)
        203  TRAV(I)=USERCST(I)
      *
            OBF=VAL + CONST
            CALL SECOND(TTOUT)
            TTOUT=TTOUT-TBEGIN
            PRINT 3000,NUMCYC,OBF,SUM,TTOUT
      *
      *  STOPPING CRITERION  CHECKING
            IF(ABS(SUM).GT.EPSI)GOTO 210
            PRINT 3004,EPSI
            GOTO 300
      *
      * CHECK THE NUMBER OF ALLOWABLE CYCLES
        210 IF(NUMCYC.LT.MAXCYC)GOTO 220
            PRINT 1003,MAXCYC
            GOTO 300
      *
      * CHECK THE RUNNING TIME
        220 NUMCYC=NUMCYC + 1
            IF(TTOUT.LT.TIMEMAX)GOTO 200
            PRINT 1002,TIMEMAX
      *
      *
        300 CONTINUE
            PRINT 2003,TITLE
            PRINT 1013,EPSI,TIMEMAX,MAXCYC
            PRINT 1019,(RUNSEC(I),I=1,MXRUNS)
      *
      *--------CALCULATE TOTAL VEHICLE HOURS AND MILES
            VEHRS = 0.0
            VEMLS = 0.0
            DO 360 I=1,NLINK
      *--------IGNORE DUMMY LINKS
            IF (MOD(NULACO(I),1000) .EQ.13) 360,352
        352 VEHRS = VEHRS + VCUM(I) * TRAV(I)
            VEMLS = VEMLS + VCUM(I) * DIST(I)
        360 CONTINUE
            VEHRS = VEHRS / 60.
            VEMLS = VEMLS / 100.
            VESPD = VEMLS / VEHRS
      *
            PRINT 1014,VEHRS,VEMLS,VESPD
      *
            IF(IPRT.LE.0)GOTO 100
      *
      *------PRINT OUT THE CALCULATED LINK FLOWS AND TRAVEL TIMES
      *
```

```
      PROGRAM TRAFFIC    73/173  OPT=1                          FTN 4.6+460        21/05/

            317 REWIND NETFIL
230             REWIND SOLFIL
                READ (NETFIL,4001) (LINE(I),I=1,10)
                WRITE(SOLFIL,4001) (LINE(I),I=1,10)
                NLINE = 45
                NPAGE = 1
235             DO 90 N = 1,NLINK
               *--------PRINT PAGE HEADING EVERY 45 LINES
                IF (NLINE .GE.45) 80,82
             80     PRINT 2000,NPAGE,TITLE
                    NLINE = 0
240                 NPAGE = NPAGE + 1
               *.
             82 NLINE = NLINE + 1
                READ (NETFIL,2004) LINE
                ADIST = IDIST / 100.0
245             ANULA = NULA / 10.0
                SPEED = ISPEED / 10.0
                TOBS = ITOBS / 100.0
                TPRED = TRAV(N)
                IVPRED = VCUM(N)
250             IF (IVOBS .GT. 0) GO TO 85
                    IVOBS = 8H   NA
                    PRINT 2006,N,NORIG,NDEST,ADIST,ANULA,DEFUNC,SPEED,TOBS,
                   +                IVOBS,LTYPE,TPRED,IVPRED
                    GO TO 90
255          85 PRINT 2005,N,NORIG,NDEST,ADIST,ANULA,DEFUNC,SPEED,TOBS,
                  +                 IVOBS,LTYPE,TPRED,IVPRED
             90 CONTINUE
            100 WRITE(SOLFIL,2011) (VCUM(I),I=1,NLINK)
                WRITE(SOLFIL,2012) (TRAV(I),I=1,NLINK)
260             WRITE (SOLFIL,2013) (KD(I),I=1,NCENT)
                REWIND SOLFIL
                END
```

```
      SUBROUTINE MINTREE(NN)
*
*       MINTREE FINDS THE SHORTEST PATHS TO ALL NODES
*       FROM THE ROOT "NN" FOR THE NETWORK DEFINED
*       BY THE ARRAYS "NOLAB", "ND" AND "TRAV".
*       ITS RETURNS THE SHORTEST PATHS IN THE ARRAY
*       "LPR".ALL THE VARIABLES IN MINTREE ARE REAL.
*
*
*       VARIABLE   LENGTH IF
*       NAME       AN ARRAY                MEANING
*       --------   ---------               -------
*
*       NNODE                                NUMBER OF NODES IN THE
*                                            NETWORK.
*       NN                                   THE ROOT NODE.
*       XINF,INF                             POSITIVE CONSTANT REPRESENTING
*                                            INFINITY
*       NLINK                                NUMBER OF LINKS(ARCS).
*
*       ND         NO.OF ARCS                LIST OF TO-NODES OF THE
*                   (NLINK)                  ARCS IN THE NETWORK IN
*                                            BY FROM NODE.
*       TRAV       NO. OF ARCS               ARC LENGTHS FOR THE ARCS
*                   (NLINK)                  IN ND.
*       NOLAB      NNODE+1                   ENTRY POINTS INTO ND
*                                            AND "TRAV"; "NOLAB(I)"
*                                            POINTS TO THE FIRST ARC
*                                            OUT OF NODE I FOR I=1,...
*                                            NNODE;
*                                            NOLAB(NNODE+1) IS 1 PLUS
*                                            THE NUMBER OF ARC.
*       NPR        NNODE                     PREDECESSORS OF THE NODES
*                                            IN THE SHORTEST PATH TREE.
*       LPR          NNODE                   LPR(I)IS THE ARC
*                                             ( NPR(I) , I )
*       D          NNODE                     LENGTHS OF THE SHORTEST
*                                            PATHS TREE.
*       CL         NNODE                     QUEUE OF NODES TO BE SCAN-
*                                            NED.
*
*
*
*          DECLARATIONS
*
*
      COMMON /NET1/ NNODE,NLINK,NCENT,NACTIV,KD(150),NOLAB(1300),
     +              ND(3000),NULACO(3000),DIST(3000)
      COMMON /NET2/ VCUM(3000),TRAV(3000), FLOW(3000),ODIJ(150)
      COMMON /STAT/ RUNSEC(10), ITERAT(10)
      COMMON /MIN/ LPR(1301),NPR(1301)
      EQUIVALENCE (RDSEC,RUNSEC(1)),(ISLSEC,RUNSEC(2)),
     +            (PRTSEC,RUNSEC(3)),(GSMSEC,RUNSEC(4)),
     +            (PTHSEC,RUNSEC(5))
*
*
      INTEGER CL(1301)
      REAL D(1301)
```

```
      SUBROUTINE MINTREE    73/173  OPT=1                    FTN 4.6+460        21/06/
                     *       INITIALISATIONS
                     *
           60              INF=3001
                           XINF=1.0E12
                           CALL SECOND(TTIN)
                     *
                           DO 100 I=1,NNODE
           65        *
                     *          SET DISTANCES TO INFINITY,CLEAR
                     *          PREDESSORS AND QUEUE.
                     *
                           D(I)=XINF
           70              NPR(I)=NN
                           LPR(I)=0
                           CL(I)=0
                     *
                     100 CONTINUE
           75        *          SET THE DISTANCE OF ROOT TO 0.
                     *
                           D(NN)=0
                     *          SET QUEUE TO COUNTAIN ONLY ROOT NN.
                     *
           80              CL(NN)=INF
                     *
                           I=NN
                           NT=NN
                     *
           85        ******
                     *     * MAIN LOOP OF ALGORITHM
                     ******
                     *
                     *          SCAN THE ARCS OUT OF NODE I - ARCS NOLAB(I)
           90        *          TO NOLAB(I+1)-1
                     *
                     120 INOLAB=NOLAB(I+1)-1
                           XD=D(I)
                           INOLAB1=NOLAB(I)
           95        *
                           IF(INOLAB1.GT.INOLAB) GOTO 201
                     *
                           DO 200 IR=INOLAB1,INOLAB
                           K=ND(IR)
          100        *
                     *          CHECK FOR DECREASE IN SHORTEST DISTANCE
                     *          IF NOT IMPROVING,GET NEXT ARC
                     *
                        32 XDIST=TRAV(IR)+XD
          105           332 IF(XDIST.GE.D(K)) GOTO 200
                     *
                     *          DECREASE OF SHORTEST DISTANCE WITH ARC (I,K)
                     *          UPDATE PREDECESSOR AND SHORTEST DISTANCE.
                     *
          110              D(K)=XDIST
                           LPR(K)=IR
                           NPR(K)=I
                     *
                     *          CHECK QUEUE STATUS OF NODE K.
```

```
      *
            IF(CL(K)) 160,140,200
      *
      *            NO PATH IS ALOWED TO PASS THROUGH A
      *             CENTROID(ORIGIN OR DESTINATION)
      *
        140 IF(K.LE.NCENT) GOTO 200
      *
      *
      *
      *            NODE K HAS NEVER BEEN SCANNED;
      *            ADD NODE K TO END OF QUEUE AFTER
      *            NODE NT
      *
        141 CL(NT)=K
      *            MAKE NODE K NEW END OF QUEUE NT AND
      *            FLAG AS END OF QUEUE.
            NT=K
            CL(K)=INF
            GOTO 200
      *            NODE K HAS ALREADY BEEN SCANNED.ADD
      *
      *            NODE K TO BEGINNING OF QUEUE JUST
      *            AFTER NODE I.
      *
        160 CL(K)=CL(I)
            CL(I)=K
      *
      *            NODE K IS IN QUEUE
      *
        200 CONTINUE
      *
      *           GET NEXT NODE I FOR BEGINNING OF QUEUE.
      *
        201 ICL=CL(I)
      *            FLAG NODE I AS HAVING BEEN SCANNED.
      *
            CL(I)=-1
            I=ICL
            IF(I.LT.INF) GOTO 120
      *
      ******
      *    * END OF THE ALGORITHM
      ******
      *
            CALL SECOND(TTOUT)
            PTHSEC=PTHSEC+TTOUT-TTIN
            RETURN
            END
```

SYMBOLIC REFERENCE MAP (R=1)

```
FUNCTION SUMTRAV    73/173  OPT=1                          FTN 4.6+460        21/06/78

      1              REAL FUNCTION SUMTRAV(LAMBDA)
                    *
                    *--------CALCULATE OBJECTIVE FUNCTION (SUM TRAVEL TIME) FOR A
                    *
      5             *        GIVEN LAMBDA.   (FUNCTION Z IN ALGOITHM STEP 4)
                     COMMON /NET1/ NNODE,NLINK,NCENT,NACTIV,KD(150),NOLAB(1300),
                    *              ND(3000),NULACO(3000),DIST(3000)
                     COMMON /NET2/ VCUM(3000),TRAV(3000), FLOW(3000),ODIJ(150)
                     COMMON        WKMEM(4300)
     10              COMMON /PARAM/ EPSI , TIMEMAX, MAXCYC, MAXIT, IPCH, IPRT,IPOST
                     COMMON /STAT/ RUNSEC(10), ITERAT(10)
                     COMMON/COEF/  ALPHA(20),BETA(20),GAMMA(20),DELTA(20)
                     DIMENSION SPEED(3)
                     REAL LAMBDA
     15             *
                     DATA SPEED/10.0,15.0,60.0/
                    *
                     SUM=0.0
                     DO 100 J=1,NLINK
     20              VOL=VCUM(J) + LAMBDA*FLOW(J)
                    *--------UNPACK DELAY CURVE NUMBER  JC
                    *               NUMBER OF LANES     NULA
                     JC=MOD(NULACO(J),1000)
                     NULA=NULACO(J)/1000
     25              ULA=NULA/10.00
                    * IT IS ASSUMED THAT VOL IS ECUALLY DISTRIBUTED OVER THE LANES
                     IF(JC.LE.10.OR.JC.GT.13)GOTO 3
                         J10=JC-10
                         SUM=SUM+VOL*0.6*DIST(J)/SPEED(J10)
     30                  GOTO 100
                   3 CONTINUE
                     GAM=GAMMA(JC)*ULA
                     DEL=DELTA(JC)* ULA
                     BET=BETA(JC)*ULA*ULA
     35              VMG=VOL-GAM
                     AVMG=VMG*ALPHA(JC)
                     RAC=SQRT(AVMG*AVMG + BET)
                     A1=VOL*(DEL+ALPHA(JC)*(VOL*0.5 - GAM))
                     A2=0.5*(VMG*RAC +BET/ALPHA(JC)*ALOG(AVMG +RAC))
     40              SUM=SUM+0.01 * DIST(J)*(A1+A2)/ULA
                 100 CONTINUE
                     SUMTRAV=SUM
                     RETURN
                     END
```

```
      FUNCTION USERCST    73/173  OPT=I                    FTN 4.6+460      21/05/
 1                REAL FUNCTION USERCST( J )
                  *
                  *--------CALCULATE USER COST = TRAVEL TIME FOR LINK J
                  *
 5                COMMON /NET1/ NNODE,NLINK,NCENT,NACTIV,KD(150),NOLAB(1300),
                 +              ND(3000),NULACO(3000),DIST(3000)
                  COMMON /NET2/ VCUM(3000),TRAV(3000), FLOW(3000),ODIJ(150)
                  COMMON /COEF/ALPHA(20),BETA(20),GAMMA(20),DELTA(20)
                  DIMENSION SPEED(3)
10                *
                  DATA SPEED/10.0,15.0,60.0/
                  *
                  VOL=VCUM(J)
                  *--------UNPACK DELAY CURVE INDEX JC
15                *              NUMBER OF LANES      NULA
                  JC=MOD(NULACO(J),1000)
                  NULA=NULACO(J)/1000
                  ULA=NULA/10.00
                  VOL=VOL/ULA
20                IF(JC.LE.10.OR.JC.GT.13)GOTO 3
                      J10=JC-10
                      USERCST=DIST(J)*0.6/SPEED(J10)
                      RETURN
                  3 VMG=VOL-GAMMA(JC)
25                AVMG=ALPHA(JC)*VMG
                  USPEED=DELTA(JC)+AVMG+SQRT(AVMG*AVMG+BETA(JC))
                  USERCST=USPEED*DIST(J)*0.01
                  RETURN
                  END
```

SUBROUTINE GSM 73/173 OPT=1 FTN 4.6+460 21/06/

```
      SUBROUTINE GSM(LAMBDA,DELTA,ACCUR,VAL)
*
*     - GOLDEN SECTION METHOD -  SEARCH
*
*        FIND LAMBDA IN THE INTERVAL ZERO TO "DELTA" SUCH THAT
*         THE OBJECTIVE FUNCTION (SUMTRAV) IS MINIMUM.
*        VAL IS SET TO THE CORRESPONDING OBJECTIVE FUNCTION
*            VALUE.
*        ACCUR IS THE REQUIRED ACCURACY, THE SEARCH TERMINATES
*              WHEN THE INTERVAL BETWEEN THE LOWER AND UPPER BOUNDS
*              OF LAMBDA IS WITHIN THE REQUIRED ACCURACY.
*
      REAL LAMBDA
*
      TAU=(1.0+SQRT(5.))*.5
      G=0.0
      D=DELTA
      XB=D/TAU
      XA=D-XB
      FXA=SUMTRAV(XA)
      FXB=SUMTRAV(XB)
   77 IF(FXA-FXB)80,80,78
   78    G=XA
         XA=XB
         XB=G+D-XA
         FXA=FXB
         FXB=SUMTRAV(XB)
         GOTO 82
   80 D=XB
      XB=XA
      XA=G+D-XB
      FXB=FXA
      FXA=SUMTRAV(XA)
   82 IF(D-G-ACCUR)83,83,77
*
   83 LAMBDA=XB
      VAL=FXB
      IF(FXB.LT.FXA)RETURN
      VAL=FXA
      LAMBDA=XA
      RETURN
      END
```

```
SUBROUTINE NETWORK    73/173  OPT=1                        FTN 4.6+460          21/06/7

      SUBROUTINE NETWORK
*
*--------READ IN NETWORK DATA FOR ALL LINKS
*
      COMMON /NET1/ NNODE,NLINK,NCENT,NACTIV,KD(150),NOLAB(1300),
     +              NC(3000),NULACO(3000),DIST(3000)
      COMMON /NET2/ VCUM(3000),TRAV(3000), FLOW(3000),ODIJ(150)
      COMMON /PARAM/ EPSI , TIMEMAX, MAXCYC, MAXIT, IPCH, IPRT,IPOST
      COMMON /STAT/ RUNSEC(10), ITERAT(10)
      COMMON /COEF/  ALPHA(20),BETA(20),GAMMA(20),DELTA(20)
      COMMON /EDIT/LINE(11),L1(26),L2(26),TITLE(10)
      COMMON /UNIT/ CARD,ODFIL,NETFIL,SOLFIL
      INTEGER CARD,ODFIL,NETFIL,SOLFIL
      DIMENSION JDEST(150),JDEM(150)
      INTEGER DEFUNC
      EQUIVALENCE (NO,LINE(1)),(NDEST,LINE(2)),(IDIST,LINE(3)),
     +           (NULA,LINE(4)),(DEFUNC,LINE(5))
*
      DATA MAXERR/3/
*
 1000 FORMAT(10A8)
 1001 FORMAT(A5,F10.0,F5.0,10I5)
 1002 FORMAT(A5,15I5)
 1003 FORMAT(5X,3(I4,1X),I3,I2,1X,I4,1X,2(I5,1X),3(I3,1X))
 1010 FORMAT (A5,4A10/A5,6A10)
 2001 FORMAT(//,5X,*E03 LINKS NOT SORTED. LINE :*,I4)
 2002 FORMAT(/5X,*E04 NO. CARDS .NE. NLINKS *,2I8)
 2051 FORMAT (5X,*E11 TITLE CARDS MISSING : *,A5,5X,A5)
 2052 FORMAT (5X,*E12 PARAMETER CARD MISSING : *,A5)
 2053 FORMAT (5X,*E13 SIZE CARD MISSING : *,A5)
 2060 FORMAT (5X,*E20 TOO MANY INPUT ERRORS :*,I5)
 3001 FORMAT(2I5,10(I3,I4))
 3011 FORMAT(/,10X,*E05 ERROR IN INPUT OD MATRIX. TYPE:*,I3,
     +       *  CARD :*,I5,*  ORIGIN :*,I5)
 4001 FORMAT(/10X,*I07 PARTIAL SOLUTION INPUT*,I3)
 4004 FORMAT(10F8.0)
*
      ERR=0
      ICARD=0
      READ 1010,ID,(TITLE(I),I=1,4),ID2,(TITLE(I),I=5,10)
      IF (ID .NE. 5HTIT1  .OR. ID2 .NE. 5HTIT2  ) 10,12
   10    PRINT 2051,ID,ID2
         IERR = IERR + 1
   12 READ 1000,(LINE(I),I=1,10)
      REWIND NETFIL
      WRITE(NETFIL,1000) (LINE(I),I=1,10)
*
*--------READ ALGORITHM CONTROL PARAMETERS
*
      READ 1001, ID,EPSI,TIMEMAX,MAXCYC,MAXIT,IPOST,IPCH,IPRT
      IF (ID .NE. 5HPARM  ) 20,22
   20    PRINT 2052,ID
         IERR = IERR + 1
   22 IF(EPSI.LE.0.0) EPSI=0.05
      IF(TIMEMAX.LE.0.0) TIMEMAX=150.0
      IF(MAXCYC.LE.0) MAXCYC=10
      MAXCYC=1
```

```
SUBROUTINE NETWORK    73/173  OPT=1                          FTN 4.6+460        21

            IF(MAXIT.LE.0) MAXIT=15
      *
      *---------READ NETWORK DATA AND CREATE NETWORK FILE
      60    READ 1002, ID,NNODE,NLINK,NCENT
            IF (ID .NE. SHSIZE  ) 30,32
         30     PRINT 2053,ID
                IERR = IERR + 1
      65 32 IF (IERR .GE. MAXERR) 34,36
         34     PRINT 2060,IERR
                CALL EXIT
      *
         36 DO 50 I=1,NNODE
      70 50     NOLAB(I)=0
            NOLAB(1)=1
            MNO=0
            I=0
            J=1
      75 55 I=I+1
      C---------READ LINK RECORD
            READ 1003,LINE
            IF (EOF(CARD)) 59,56
         56 ND(I)=NDEST
      80    DIST(I)=IDIST
            MNO=MAXO(NO,NDEST,MNO)
            NULACO(I) = NULA * 1000 + DEFUNC
            WRITE(8,1003) LINE
      *---------SET UP LINK ORIGIN INDEX TABLE NOLAB
      85    IF(NO-J)57,55,58
         57     PRINT 2001,I
                CALL EXIT
         58 NOLAB(NO)=I
            J=NO
      90    GOTO 55
      *
         59 I=I-1
            IF(I.NE.NLINK)60,61
         60     PRINT 2002,I,NLINK
      95        CALL EXIT
         61 CONTINUE
      *
            NNODE = MNO
            NOLAB(NNODE+1) = NLINK + 1
     100    IX1=NLINK+1
            DO 2 I=1,NNODE
            II=NNODE+1-I
            IX2=NOLAB(II)
            IF(IX2.NE.0) GOTO 1
     105    NOLAB(II)=IX1
            GOTO 2
          1 IX1=IX2
          2 CONTINUE
            NACTIV=0
     110    NOLD=0
            DO 102 IO=1,NCENT
      *
      *      READ IN THE DEMANDS ASSOCIATED WITH THE CURRENT ORIGIN
      *
```

```
      SUBROUTINE NETWORK    73/173  OPT=1                    FTN 4.6+460        21/05/7
C
C     KD WILL CONTAIN ORIGIN NODE NUMBER OF EACH ORIGIN IN THE
C        OD MATRIX IN THE ORDER READ IN. KD ENTRY WILL BE NEGATIVE
C        IF THERE IS NO DEMAND FROM THAT ORIGIN.
C
      DO 101 I=1,NCENT
  101     ODIJ(I)=0.00
      M=10
      ICARD=ICARD+1
      READ 3001,N,NT,(L1(J),L2(J),J=1,10)
      IF (EOF(CARD)) 105,106
  105     IERR = 2
          GO TO 113
  106 IF(N.EQ.0.OR.N.EQ.NOLD)111,112
  111     IERR=1
  113     PRINT 3011,IERR,ICARD,IO
          CALL EXIT
  112 KD(IO)=-N
      NOLD=N
C--------IF NO DEMAND FOR THIS ORIGIN KD ENTRY NEGATIVE
      IF(NT.EQ.0)GOTO 102
      KD(IO)=N
C--------SET UP DEMAND MATRIX ODIJ FOR THIS ORIGIN
      DO 120 I=1,10
      JDEST(I)=L1(I)
  120 JDEM(I)=L2(I)
      IF(NT.LE.10)GOTO 150
      NUM=NT/10
      IF(NUM*10.EQ.NT) NUM=NUM-1
      DO 121 I=1,NUM
      ICARD=ICARD+1
      READ 3001,N,NT,(L1(J),L2(J),J=1,10)
      IF (EOF(CARD)) 105,126
  126 IF(N.NE.NOLD)GOTO 111
      DO 122 J=1,10
      M=M+1
      JDEST(M)=L1(J)
  122 JDEM(M)=L2(J)
  121 CONTINUE
C
C
  150  CONTINUE
      DO 116 J=1,NT
      IF(JDEST(J).EQ.N)GOTO 116
      DEM=JDEM(J)
      NEXT=JDEST(J)
      ODIJ(NEXT)=DEM
      NACTIV=NACTIV+1
  116 CONTINUE
      WRITE(ODFIL) ODIJ
  102 CONTINUE
C
      RETURN
C
C
C      RE-ESTABLISH PARTIAL SOLUTION
C
      ENTRY PARTSOL
```

```
SUBROUTINE NETWORK    73/173  OPT=1                    FTN 4.6+460        21/06/79

                      PRINT 4001,IPOST
                      REWIND SOLFIL
                      READ(SOLFIL,1000) (LINE(I),I=1,10)
175                   PRINT 1000,(LINE(I),I=1,10)
                      READ(SOLFIL,4004) (VCUM(I),I=1,NLINK)
                      DO 206 I=1,NLINK
                  206 TRAV(I)=USERCST(I)
                      RETURN
180                   END
```

```
SUBROUTINE LOADTR     73/173  OPT=1                    FTN 4.6+460        21/06/79

      SUBROUTINE LOADTR(NN)
*
*---------ACCUMULATE FLOW ON LINKSUSED ON ROUTE TO ALL DESTINATIONS
*         FROM THE GIVEN ORIGIN (NN)
      COMMON /NET1/ NNODE,NLINK,NCENT,NACTIV,KD(150),NOLAB(1300),
     +              ND(3000),NULACO(3000),DIST(3000)
      COMMON /NET2/ VCUM(3000),TRAV(3000), FLOW(3000),ODIJ(150)
      COMMON /STAT/ RUNSEC(10), ITERAT(10)
      COMMON /MIN/ LPR(1301),NPR(1301)
*
*---------CONSTRUCT SHORTEST PATHS FROM ORIGIN NN TO ALL DESTINATIONS.
C
      CALL MINTREE(NN)
      LPR(NN)=0
C
*
      DO 30 I=1,NCENT
      TRIP=ODIJ(I)
      IF(TRIP.LE.0.0) GOTO 30
      NEXT=I
   40 L=LPR(NEXT)
      IF(L.EQ.0) GOTO 30
      FLOW(L)=FLOW(L)+ TRIP
      NEXT=NPR(NEXT)
      GOTO 40
   30 CONTINUE
*
      RETURN
      END
```

```
SUBROUTINE EXFLOW     73/173  OPT=1                    FTN 4.6+460        21/06/78

      1               SUBROUTINE EXFLOW
                C..
                      COMMON /NET1/ NNODE,NLINK,NCENT,NACTIV,KD(150),NOLAB(1300),
                     +              ND(3000),NULACO(3000),DIST(3000)
      5               COMMON /NET2/ VCUM(3000),TRAV(3000), FLOW(3000),ODIJ(150)
                      COMMON /STAT/ RUNSEC(10), ITERAT(10)
                      COMMON /UNIT/ CARD,ODFIL,NETFIL,SOLFIL
                      COMMON /MIH/ LPR(1301),MPR(1301)
                      EQUIVALENCE (RDSEC,RUNSEC(1)),(ISLSEC,RUNSEC(2)),
     10               +            (PRTSEC,RUNSEC(3)),(GSMSEC,RUNSEC(4)),
                      +            (PTHSEC,RUNSEC(5))
                      INTEGER CARD,ODFIL,NETFIL,SOLFIL
                C
                      REWIND ODFIL
     15               *
                      DO 190 I=1,NLINK
                  190 FLOW(I)=0.0
                C
                C
     20               DO 42 N=1,NCENT
                          NN=KD(N)
                          IF(NN.LE.0)GOTO 42
                      *--------READ DEMAND FOR THIS ORIGIN
                          READ (ODFIL) ODIJ
     25               *--------ACCUMULATE ALL OR NOTHING SOLUTION (FLOW) FOR THIS ORIGIN
                          CALL LOADTR(NN)
                     42   CONTINUE
                      RETURN
                      END
```

ANNEXE C

Programme utilisé pour le Réseau de la ville de Hull

PROGRAM TACL2 73/173 OPT=1 FTN 4.6+460 21/06/78 09.23

```
      PROGRAM TACL2(INPUT,OUTPUT,TAPE1,TAPE3,TAPE5,TAPE6,TAPE9)
      COMMON/UNIT/CURNET,TABIN,LGFIL
      DIMENSION IDEN(10)
      INTEGER CURNET,TABIN
      DATA CURNET,TABIN,LGFIL /1,3,9/
*
*
      READ 1004,(IDEN(J),J=1,8)
      PRINT 1004,(IDEN(J),J=1,8)
 1004 FORMAT (5X,7A8,A4)
*
      CALL TRAFIC(IDEN)
*
      END
```

```
SUBROUTINE TRAFIC     73/173  OPT=1                    FTN 4.6+460        21/06/78

      1               SUBROUTINE TRAFIC (IDEN)
                      C
                      CC
                      C
                      COMMON/UNIT/CURNET,TABIN,LGFIL
      5               COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)
                      COMMON/OD1/NCENT,GPQ(30,30),UPQ(30,30),KORIG(30)
                      COMMON /STAT/ RUNSEC(10)
                      COMMON /PARAM/ EPSI,TIME, MAXIT,PRINT,DEMAND,LIMKEEP
     10               COMMON /GAMAS/GAMMA(35)
                      COMMON /INDEXMS/ IX(36),KEY
                      COMMON /LDATA/TO(400),BETA(400),LTYP(400)
                      DIMENSION IDEN(8),FLOW(400)
                      EQUIVALENCE (RDSEC,RUNSEC(1)),(SOLSEC,RUNSEC(2)),
     15              +            (SHSEC,RUNSEC(3)),(PTHSEC,RUNSEC(4))
                      INTEGER CURNET,TABIN
                      REAL LWBND,LWBND1
                      INTEGER U
                      REAL LAMBDA,NEWSUM
     20               LOGICAL AWAY,FROMST3
                      DATA EPSI,TIME,MAXIT/0.0001,200.0,50/
                    C
                     1002 FORMAT(//1H0,20X,"6102I AFFECTATION TERMINEE.....DUREE (SECS)",
                    +       " MAXIMUM :", F10.0)
     25              1003 FORMAT(//1H0,20X,"6103I AFFECTATION TERMINEE ... ITERATION",
                    +       " MAXIMUM :",I8)
                     1019 FORMAT ("0 SECS CPU UTILISE POUR - ",
                    +        "ENTRER LES DONNEES DU RESEAU",T60,"=",F10.2/
                    +    T24," - ETABLIR LA SOLUTION INITIALE",T60,"=",F10.2/
     30             +    T24," - CALCULER LAMBDA (TOTALE)",T60,"=",F10.2/
                    +    T24," - TROUVER LES CHEMINS PLUS COURTS",T60,"=",F10.2)
                     2019 FORMAT (7A8,A4,1X,"IT=",I3,A10)
                     2999 FORMAT(1H0,20X,
                    +          "F O N C T I O N   O B J E C T I V E *,
     35             +/,3X,"ITERATION       VALUE              (LB-VALUE)X",
                    +5X,"SEC. CUMUL"/)
                     3000 FORMAT(3X,I3,5X,E16.10,7X,F7.4,5X,F5.2)
                    C
                    *
     40             *       ---- STEP 0 ----  ( INITIALISATIONS )
                    *
                    *
                    *
                          CALL OPENMS(LGFIL,IX,36,0)
     45             *                  PLUS GRANDE BORNE INFERIEURE SUR LA FONCTION
                    *                  OBJECTIVE
                          LWBND=-1.0E99
                    *                  RETIENT L'AMELIORATION DE L'OBJECTIF LORS DE
                    *                    LA DERNIERE ITERATION REGULIERE
     50             *
                          DELTARG=0.0
                    *                  RETIENT L'AMELIORATION DE L'OBJECTIF LORS DU
                    *                  DU DERNIER AWAY OU TOWARD STEP
                          DELTAAT=1.0E99
     55             *                  NOMBRE DE POINTS EXTREMES CONSERVES INITIALEMENT - 1
                          LIMKEEP=8
                    *                  LIMITE MAXIMALE DE POINTS EXTREMES A CONSERVER
```

```
      SUBROUTINE TRAFIC     73/173  OPT=1                          FTN 4.6+460        21/06/78
                  MAXKEEP=35
                  GAMMA(1)=1.0
   60       C
                  DO 10 I=1,10
               10  RUNSEC(I)=0.0
                  CALL SECOND (TBEGIN)
                  IPAGE =1
   65       C
            C--------READ IN NETWORK
            C
                  CALL RDNET (CURNET,NCENT)
            C
   70       *------READ TRIP MATRX OR UPO MATRIX
            *
                  CALL RDOD (TABIN,NCENT,GPO,KORIG)
               60 CALL SECOND(TTOUT)
                  RDSEC=TTOUT-TBEGIN
   75             TTIN=TTOUT
            C
            C--------SET UP INITIAL (ALL OR NOTHING) SOLUTION
                  DO 55 J=1,NLINK
                  VCUM(J)=0.0
   80             FLOW(J)=0.0
               55 TRAV(J)=UCOST(J,VCUM(J))
            C
                  NUMCYC=1
                  CALL EXFLOW(FLOW)
   85             DO 5100 J=1,NLINK
                  VCUM(J)=FLOW(J)
            5100 TRAV(J)=UCOST(J,VCUM(J))
                  CALL  WRITMS(LGFIL,FLOW,400,NUMCYC)
            *
   90       *         ENREGISTRE LA SOLUTION INITIALE
            *
                  CALL SECOND(TTOUT)
                  SOLSEC=TTOUT-TTIN
            C
   95             TTIN=TTOUT
            C
            C--------PRINT HEADING FOR ITERATION LOG
              100 IPAGE = IPAGE + 1
                  PRINT 2999
  100             CALL PRINTS
                  CALL OBFJN(FVAL,VCUM,NLINK)
                  PRINT 987,NUMCYC,FVAL
                  U=1
            *
  105       *
                  CALL SECOND(TINIT)
            *
            *
            *         ---- STEP 1 ----    ( REPREND POUR L POINTS EXTREMES)
  110       *
            *
            10000 NUMCYC=NUMCYC+1
            *
            *
```

```
      SUBROUTINE TRAFIC     73/173  OPT=1                          FTN 4.6+460        21/09/73

               10101 CONTINUE
                     *
                     *
     175             *          MISE A JOUR DE L'AMELIORATION DE L'OBJECTIF
                     *
                     *
                           ABC=FVAL
                           CALL OBFJN(FVAL,VCUM,NLINK)
     180                   IF(U.EQ.2.AND..NOT. AWAY) DELTAAT= ABC-FVAL
                           IF(U.EQ.1) DELTARG=ABC-FVAL
                     *
                     *
                           XYZ=NEWSUM/FVAL
     185                   CALL SECOND(TEMPS)
                           TEMPS=TEMPS-TINIT
                     *
                     *
                           PRINT 987,NUMCYC,FVAL,XYZ,LWBND,TEMPS
     190             987 FORMAT(10X,I4,4X,E20.7,4X,F7.4,' LB: ',E20.7,' TEMPS:',F9.5)
                     *
                           IF(U.EQ.2) GOTO 10102
                     *
                     *          MISE A JOUR DES GAMMAS POUR UNE ITERATION REGULIERE
     195             *
                           IF(NUMCYC.GT.LIMKEEP) GOTO 11002
                           CALL UPDATG(NUMCYC,LAMBDA)
                           GOTO 10104
               11002 CALL UPDATG(LIMKEEP,LAMBDA)
     200                   GOTO 10104
                     *
                     *
                     *          MISE A JOUR DES GAMMAS POUR AWAY OU TOWARD STEP
                     *
     205             10102 IF(AWAY) GOTO 10103
                           CALL UPDATGT(KEYKEEP,LAMBDA,LIMKEEP)
                           GOTO 10104
               10103 CALL UPDATGA(KEYKEEP,LAMBDA,LIMKEEP,TETA)
                     *
     210             *
                     *          CONTROLE SI ON EST DANS UNE PHASE DE RAFINEMENT
                     *          DELA LINEARISATION INTERNE DU DOMAINE(AJOUTE UN
                     *          POIT EXTREME) OU UNE PHASE RESTRICTIVE( AWAY-TOWARD)
                     *
     215             *
               10104 IF(NUMCYC.LT.LIMKEEP) GOTO 10000
                     *
                     *
                     *          RETOUR POUR UN AWAY OU TOWARD STEP
     220             *
                         U=2
                           IF(DELTAAT.GE.DELTARG) GOTO 10050
                           NUMCYC=NUMCYC+1
                     *
     225             *
                     *       ---- STEP 9 ---- (GENERE UN POINT EXTREME ADDITIONNEL
                     *                          S'IL EST PERMIS DE CONTINUER)
```

```
   SUBROUTINE TRAFIC     73/173  OPT=1                          FTN 4.6+460        21/06/73

                *
230             *
                10200 IF(NUMCYC.GT.MAXIT) GOTO 5001
                      LIMKEEP=LIMKEEP+1
                      IF(LIMKEEP.GE.MAXKEEP) GOTO 500
                *
235             *
                      PRINT 9876,NUMCYC,LIMKEEP
                 9876 FORMAT(* AJOUTE UN POINT EXTREMAL, CYCLE : *,I4,
                     +* LIMKEEP : *,I4)
                *
240             *     U=1
                *
                      CALL EXFLOW(FLOW)
                      CALL WRITMS(LGFIL,FLOW,400,LIMKEEP)
                *
245             *     CALL SUMS(NEWSUM,FLOW)
                *
                      CALL OBFJN(FVAL,VCUM,NLINK)
                *
                *          CONSERVE LA PLUS GRANDE BORNE INFERIEURE POTENTIELLE
250             *
                      LWBND1=FVAL+NEWSUM
                      IF(LWBND1.GE.LWBND) LWBND=LWBND1
                *
                *          CRITERE D'ARRET
255             *
                      IF(ABS((FVAL-LWBND)/FVAL).LT.EPSI) GOTO 500
                      GOTO 10100
                C
                *
260               499 PRINT 1449,MAXKEEP
                 1449 FORMAT(*  ESPACE MEMOIRE DEMANDE INSUFFISANT*,I4)
                 5001 PRINT 1003,MAXIT
                *
                C--------OPTIMUM IS REACHED OR TIME LIMIT OR CYCLE LIMIT
265             C
                  500 CONTINUE
                      IPAGE =IPAGE + 1
                      PRINT 1019,(RUNSEC(I),I=1,4)
                *
270             C
                      CALL PRINTS
                      RETURN
                      END
```

```
 SUBROUTINE SUMS       73/173  OPT=1                          FTN 4.6+460        21/06/78

 1                     SUBROUTINE SUMS(NEWSUM,FLOW)
                      *
                      *
                      *      MESURE L'AMELIORATION DE L'OBJECTIF POUR
 5                    *      LA DIRECTION DE DESCENTE X EN SUPPOSANT
                      *      QU'IL N'Y A PAS DE CONTRAINTES .
                      *
                      *
                      *
                      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)
10                    DIMENSION FLOW(400)
                      REAL NEWSUM
                      NEWSUM=0.0
                      DO 10201 J=1,NLINK
                      X=FLOW(J)-VCUM(J)
15                    FLOW(J)=X
                      NEWSUM=NEWSUM+X*TRAV(J)
                10201 CONTINUE
                      RETURN
                      END
```

```
SUBROUTINE PRINTS      73/173  OPT=1                        FTN 4.6+460        21/0

                SUBROUTINE PRINTS
                COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)
           2020 FORMAT (I4,2X,I4,E18.9,E18.9)
      *
                PRINT 99
             99 FORMAT(2(/))
                DO 90 NO=1,NNODE
                J=NOLAB(NO)
                IF(J.LE.0)GOTO 90
                II=NO
             91 II=II+1
                JJ=NOLAB(II)-1
                IF(JJ.LE.0)GOTO 91
                DO 92 N=J,JJ
      C
                PRINT 2020, NO,ND(N),VCUM(N),TRAV(N)
             92 CONTINUE
      C
             90 CONTINUE
      *
                RETURN
                END
```

```
      SUBROUTINE RDNET(INNET,NCENT)
C
C        CURNET CONTAINS THE UNIT NUMBER OF THE FILE FROM WHICH
C             THE LINK DATA WILL BE READ.
C
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)
      COMMON /LDATA/TO(400),BETA(400),LTYP(400)
*
      DATA MAXERR/3/
      DATA MAXNOD/250/
 1106 FORMAT (1H0,20X,"DIMENSIONS SPECIFIEES TROUVEES"/
     +               23X,   "- SOMM",I9,I10/
     +               23X,   "- LIEN",I9,I10/
     +               23X,   "- CENT",I9/ 1H1)
 2001 FORMAT(1H0,20X,"0601E LIENS NON TRIES. LIEN :",I4//)
C
C--------READ DIMENSION RECORD
      REWIND INNET
      READ(INNET,3000) NCENT,NNODE,NLINK
 3000 FORMAT(5I5)
 3001 FORMAT(5X,2I5,F6.0,10X,F8.0,I3)
C
   36 DO 50 I=1,MAXNOD
   50     NOLAB(I)=0
      NOLAB(1)=1
      I=0
      MNO=0
      IERR = 0
      J=1
   55 I=I+1
C--------READ LINK RECORD
      READ( INNET, 3001) NO,ND(I),TO(I),BETA(I),LTYP(I)
      IF (EOF(INNET)) 59,56,59
   56 MNO=MAX0 (NO,ND(I),MNO)
C--------SET UP LINK ORIGIN INDEX TABLE NOLAB
      IF(NO-J)57,55,58
   57     PRINT 2001,I
          CALL EXIT
   58 NOLAB(NO) = I
      J=NO
      GOTO 55
C
   59 I=I-1
  110 PRINT 1106, NNODE,MNO, NLINK,I, NCENT
      NLINK=I
      NNODE = MNO
      NACTIV=0
*
*
*      MET LE VECTEUR DE POINTEURS SOUS FORME ACCEPTABLE
*      POUR LA SOUROUTINE  MINTRE .
*
*
      NOLAB(NNODE+1) = NLINK + 1
      IX1=NLINK+1
      DO 2 I=1,NNODE
      II=NNODE+1-I
```

```
  SUBROUTINE RDNET      73/173  OPT=1                    FTN 4.6+460        21/06/73

                    IX2=NOLAB(II)
                    IF(IX2.NE.0) GOTO 1
 60 .               NOLAB(II)=IX1
                    GOTO 2
                  1 IX1=IX2
                  2 CONTINUE
                    RETURN
 65                 END
```

```
  SUBROUTINE RDOD       73/173  OPT=1                    FTN 4.6+460        21/06/73

 1                  SUBROUTINE RDOD(INFIL,NCENT,TAB,INDEX)
                    DIMENSION TAB(30,30),ROW(30),INDEX(30)
                *
                 4001 FORMAT(8F8.0)
 5              *
                    REWIND INFIL
                    DO 10 I=1,NCENT
                 10 INDEX(I)=-I
                    DO 20 I=1,NCENT
 10                 READ (INFIL,4001) (ROW(J),J=1,NCENT)
                    SUM= 0
                    DO 30 J=1,NCENT
                    TAB(I,J)=ROW(J)*.1
                    IF(I.EQ.J)TAB(I,J)=0.
 15 .               30 SUM=SUM+ROW(J)
                    IF(SUM.LE.0.0)GOTO 20
                    INDEX(I)=I
                    PRINT 4001,(ROW(J),J=1,NCENT)
                 20 CONTINUE
 20                 RETURN
                    END
```

```
SUBROUTINE LOADTR     73/173  OPT=1                    FTN 4.6+460        21/05/
      SUBROUTINE LOADTR(NN,FLOW)
C
C
C     PROCEDE A UNE AFFECTATION 'ALL NOTHING' SUR LES
C     CHEMINS LES PLUS COURTS,DE L'ORIGINE  NN  VERS
C     TOUTES LES AUTRES DESTINATIONS.
C
C
      COMMON/OD1/NCENT,GPQ(30,30),UPQ(30,30),KORIG(30)
      COMMON /STAT/ RUNSEC(10)
      EQUIVALENCE (RDSEC,RUNSEC(1)),(SOLSEC,RUNSEC(2)),
     +            (SHSEC,RUNSEC(3)),(PTHSEC,RUNSEC(4))
      DIMENSION FLOW(400)
      INTEGER LPR(250),NPR(250)
C
      CALL MINTREE(NN,NPR,LPR)
      LPR(NN)=0
C
C
      DO 30 I=1,NCENT
      TRIP=GPQ(NN,I)
      IF(TRIP.LE.0.0) GOTO 30
      NEXT=I
   40 L=LPR(NEXT)
      IF(L.EQ.0) GOTO 30
      FLOW(L)=FLOW(L)+ TRIP
      NEXT=NPR(NEXT)
      GOTO 40
   30 CONTINUE
C
      RETURN
      END
```

FUNCTION DUCOST 73/173 OPT=1 FTN 4.6+460 21/05/7

```
      REAL FUNCTION DUCOST(L,VOL)
      COMMON/LDATA/ TO(400),BETA(400),LTYP(400)
      DUCOST=0.0
      IF(LTYP(L).EQ.1.OR.LTYP(L).EQ.4)RETURN
      C=VOL/BETA(L)
      C=C*C*C
      DUCOST=TO(L)*.6*C/BETA(L)
      RETURN
      END
```

FUNCTION TCOST 73/173 OPT=1 FTN 4.6+460 21/05/7

```
      REAL FUNCTION TCOST(L,VOL)
      COMMON/LDATA/ TO(400),BETA(400),LTYP(400)
      TCOST=0.0
      IF(LTYP(L).EQ.1.OR.LTYP(L).EQ.4)RETURN
      CA2=BETA(L)*BETA(L)
      FA2=VOL*VOL
      TCOST=VOL*TO(L)*(FA2*FA2/(CA2*CA2*5.)*.15+1.)
      RETURN
      END
```

FUNCTION UCOST 73/173 OPT=1 FTN 4.6+460 21/06/7

```
      REAL FUNCTION UCOST(L,VOL)
      COMMON/LDATA/ TO(400),BETA(400),LTYP(400)
      UCOST=0.
      IF(LTYP(L).EQ.1.OR.LTYP(L).EQ.4)RETURN
      C=VOL/BETA(L)
      C=C*C*C*C
      UCOST=TO(L)*(1.+.15*C)
      RETURN
      END
```

```
 SUBROUTINE OBFJN      73/173  OPT=1                          FTN 4.6+460        21/05/7

 1                SUBROUTINE OBFJN(VAL, ARG,NLINK)
                  DIMENSION ARG(100)
            *
            *
 5                VAL=0.0
                  DO 10 I=1,NLINK
               10  VAL=VAL + TCOST(I,ARG(I))
            *
                  RETURN
10                END
```

```
 SUBROUTINE GRADY      73/173  OPT=1                          FTN 4.6+460        21/06/7

 1                SUBROUTINE  GRADY( SUM.ARG,Y,NLINK)
                  DIMENSION ARG(100),Y(100)
            *
            *
 5                VAL=0.0
                  DO 10 I=1,NLINK
               10 VAL=VAL + UCOST(I,ARG(I))*Y(I)
                  SUM=VAL
            *
10                RETURN
                  END
```

73/173 OPT=1 FTN 4.6+460 21/?

```
      SUBROUTINE MINTREE(NN,NPR,LPR)
*
*       MINTREE FINDS THE SHORTEST PATHS TO ALL NODES
*       FROM THE ROOT "NN" FOR THE NETWORK DEFINED
*       BY THE ARRAYS "NOLAB", "ND" AND "TRAV".
*       ITS RETURNS THE SHORTEST PATHS IN THE ARRAY
*       "LPR".ALL THE VARIABLES IN MINTREE ARE REAL.
*
*
*        VARIABLE    LENGTH IF
*        NAME        AN ARRAY                MEANING
*        --------    ---------               -------
*
*        NNODE                                  NUMBER OF NODES IN THE
*                                               NETWORK.
*        NN                                     THE ROOT NODE.
*        XINF,INF                               POSITIVE CONSTANT REPRESENTING
*                                               INFINITY
*        NLINK                                  NUMBER OF LINKS(ARCS).
*
*        ND          NO.OF ARCS                 LIST OF TO-NODES OF THE
*                     (NLINK)                   ARCS IN THE NETWORK IN
*                                               BY FROM NODE.
*        TRAV        NO. OF ARCS                ARC LENGTHS FOR THE ARCS
*                     (NLINK)                   IN ND.
*        NOLAB       NNODE+1                    ENTRY POINTS INTO ND
*                                               AND "TRAV"; "NOLAB(I)"
*                                               POINTS TO THE FIRST ARC
*                                               OUT OF NODE I FOR I=1,...
*                                               NNODE;
*                                               NOLAB(NNODE+1) IS 1 PLUS
*                                               THE NUMBER OF ARC.
*        NPR         NNODE                      PREDECESSORS OF THE NODES
*                                               IN THE SHORTEST PATH TREE.
*        LPR         NNODE                      LPR(I) IS THE ARC
*                                                ( NPR(I) , I )
*        D           NNODE                      LENGTHS OF THE SHORTEST
*                                               PATHS TREE.
*        CL          NNODE                      QUEUE OF NODES TO BE SCAN-
*                                               NED.
*
*
*
*          DECLARATIONS
*
      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)
      COMMON/OD1/NCENT,GPQ(30,30),UPQ(30,30),KORIG(30)
      COMMON /STAT/ RUNSEC(10)
      EQUIVALENCE (RDSEC,RUNSEC(1)),(SOLSEC,RUNSEC(2)),
     +            (SHSEC,RUNSEC(3)),(PTHSEC,RUNSEC(4))
      INTEGER LPR(250),NPR(250)
      DIMENSION FLOW(100)
      INTEGER CL(250)
      REAL D(250)
*       INITIALISATIONS
*
      INF=9999999999999
      XINF=999999999.9
```

```
SUBROUTINE MINTREE    73/173  OPT=1                    FTN 4.6+460        21/06/77

                      CALL SECOND(TTIN)
                *
60                    DO 100 I=1,NNODE
                *
                *           SET DISTANCES TO INFINITY,CLEAR
                *           PREDESSORS AND QUEUE.
                *
65                    D(I)=XINF
                      NPR(I)=NN
                      LPR(I)=0
                      CL(I)=0
                *
70                100 CONTINUE
                *           SET THE DISTANCE OF ROOT TO 0.
                *
                      D(NN)=0
                *           SET QUEUE TO COUNTAIN ONLY ROOT NN.
75              *
                      CL(NN)=INF
                *
                      I=NN
                      NT=NN
80              *
                ******
                *     * MAIN LOOP OF ALGORITHM
                ******
                *
85              *           SCAN THE ARCS OUT OF NODE I - ARCS NOLAB(I)
                *           TO NOLAB(I+1)-1
                *
              120 INOLAB=NOLAB(I+1)-1
                      XD=D(I)
90                    INOLAB1=NOLAB(I)
                *
                      IF(INOLAB1.GT.INOLAB) GOTO 201
                *
                      DO 200 IR=INOLAB1,INOLAB
95                    K=ND(IR)
                *           COMPUTE DISTANCE TO NODE K
                *           USING ARC (I,K)
                      XDIST=TRAV(IR)+XD
                *
100             *           CHECK FOR DECREASE IN SHORTEST DISTANCE
                *           IF NOT IMPROVING,GET NEXT ARC
                *
                      IF(XDIST.GE.D(K)) GOTO 200
                *
105             *           DECREASE OF SHORTEST DISTANCE WITH ARC (I,K)
                *           UPDATE PREDECESSOR AND SHORTEST DISTANCE.
                *
                      D(K)=XDIST
                      LPR(K)=IR
110                   NPR(K)=I
                *
                *           CHECK QUEUE STATUS OF NODE K.
                *
                      IF(CL(K)) 160,140,200
```

```
      SUBROUTINE MINTREE    73/173  OPT=1                          FTN 4.6+460        21/0
  115           *
                *           NO PATH IS ALOWED TO PASS THROUGH A CENTROID
                *           (ORIGIN OR DESTINATION).
                *
                  140 IF(K.LE.NCENT) GOTO 200
  120           *
                *
                *         NODE K HAS NEVER BEEN SCANNED;
                *         ADD NODE K TO END OF QUEUE AFTER
                *         NODE NT
  125           *
                  141 CL(NT)=K
                *         MAKE NODE K NEW END OF QUEUE NT AND
                *         FLAG AS END OF QUEUE.
                      NT=K
  130                 CL(K)=INF
                      GOTO 200
                *         NODE K HAS ALREADY BEEN SCANNED.ADD
                *
                *         NODE K TO BEGINNING OF QUEUE JUST
  135           *         AFTER NODE I.
                *
                  160 CL(K)=CL(I)
                      CL(I)=K
                *
  140           *         NODE K IS IN QUEUE
                *
                  200 CONTINUE
                *
                *       GET NEXT NODE I FOR BEGINNING OF QUEUE.
  145           *
                  201 ICL=CL(I)
                *         FLAG NODE I AS HAVING BEEN SCANNED.
                *
                      CL(I)=-1
  150                 I=ICL
                      IF(I.LT.INF) GOTO 120
                *
                ******
                *    * END OF THE ALGORITHM
  155           ******
                *
                      CALL SECOND(TTFIN)
                      PTHSEC=PTHSEC+TTFIN-TTIN
                      RETURN
  160                 END


      SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
    3  MINTREE
```

```
SUBROUTINE EXFLOW     73/173  OPT=1                          FTN 4.6+460        21/06/

      1               SUBROUTINE EXFLOW(FLOW)
                     *
                     *
                     *      PROCEDE A UNE AFFECTATION 'ALL NOTHING' SUR
                      5      LES CHEMINS LES PLUS COURTS,POUR TOUT COU-
                     *       PLE ORIGINE-DESTINATION.
                     *
                     *
                              COMMON/UNIT/CURNET,TABIN,LGFIL
                     10       COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)
                              COMMON/OD1/NCENT,GPQ(30,30),UPQ(30,30),KORIG(30)
                              COMMON /STAT/ RUNSEC(10)
                              COMMON /PARAM/ EPSI,TIME, MAXIT,PRINT,DEMAND,LIMKEEP
                              COMMON /GAMAS/GAMMA(35)
                     15       DIMENSION IDEN(8),FLOW(400)
                              EQUIVALENCE (RDSEC,RUNSEC(1)),(SOLSEC,RUNSEC(2)),
                             +            (SHSEC,RUNSEC(3)),(PTHSEC,RUNSEC(4))
                     *
                              DO 10 I=1,NLINK
                     20     10 FLOW(I)=0.0
                     *
                              DO 20 N=1,NCENT
                              NN=KORIG(N)
                              IF(NN.LE.0) GOTO 20
                     25       CALL LOADTR(NN,FLOW)
                            20 CONTINUE
                              RETURN
                              END
```

```
SUBROUTINE UPDATGT    73/173  OPT=1                    FTN 4.6+460         21/06/7

            SUBROUTINE UPDATGT(N,LAMBDA,LIMKEEP)
      *
      *
      *       MISE A JOUR DES VARIABLES GAMMA LORS DE
      *       L'EXECUTION D'UN TOWARD STEP.
      *
      *
      *
            COMMON /GAMAS/GAMMA(35)
            REAL LAMBDA
            PRINT 99,LAMBDA,N
         99 FORMAT(5X,* APPEL DE UPDATGT ---,LAMBDA : *,F14.9,
           +* VECTEUR GARDE NO :*,I4)
      *
            X=1.0 - LAMBDA
            GAMMA(N)=GAMMA(N)*X+LAMBDA
      *
            DO 10 J=1, LIMKEEP
            IF(J.EQ.N) GOTO 10
            GAMMA(J)=GAMMA(J)*X
         10 CONTINUE
      *
      *
            RETURN
            END
```

```
SUBROUTINE UPDATGA    73/173  OPT=1                    FTN 4.6+460        21/06/7

1               SUBROUTINE UPDATGA(N,LAMBDA,LIMKEEP,TETA)
                *
                *
                *       MISE A JOUR DES VARIABLES GAMMA LORS DE
5               *       L'EXECUTION D'UN AWAY STEP.
                *
                *
                      COMMON /GAMAS/GAMMA(35)
                      REAL LAMBDA
10              *
                      PRINT 99,LAMBDA,N
                   99 FORMAT(5X,* APPEL DE UPDATGA ###,LAMBDA : *,F14.9,
                     +* VECTEUR GARDE NO :*,I4)
                      X=1.0 - LAMBDA
15                    Y=LAMBDA*TETA
                      Z=X+Y
                      GAMMA(N)=GAMMA(N)*(X+Y)+LAMBDA*(1.0 - TETA)
                *
                      DO 10 J=1, LIMKEEP
20                    IF(J.EQ.N) GOTO 10
                      GAMMA(J)=GAMMA(J)*(Z)
                   10 CONTINUE
                *
                      RETURN
25                    END
```

```
SUBROUTINE UPDATG     73/173  OPT=1                          FTN 4.6+460        21/
      SUBROUTINE UPDATG(LIMKEEP,LAMBDA)
*
*       MISE A JOUR DES VARIABLES GAMMA LORS DE
*       L'ENREGISTREMENT D'UN NOUVEAU POINT EXTREME.
*
*
      COMMON /GAMAS/ GAMMA(35)
      REAL LAMBDA
      X=1.0 - LAMBDA
      PRINT 99,LAMBDA
   99 FORMAT(5X,* APPEL DE UPDATG  %%%,LAMBDA : *,F14.9)
      NM1=LIMKEEP-1
      DO 10 I=1,NM1
      GAMMA(I)=GAMMA(I)*X
   10 CONTINUE
*
      GAMMA(LIMKEEP)=LAMBDA
      RETURN
      END
```

```
SUBROUTINE NEWFLOW    73/173  OPT=1                    FTN 4.6+460        21/

      1               SUBROUTINE NEWFLOW(NEWSUM,AWAY,KEYKEEP,TETA,FLOW,NUMCYC)
                    *
                    *
                    *       PREPARE UNE DIRECTION DE DESCENTE BASEE SUR
                    *       UN AWAY STEP OU TOWARD STEP .LE VECTEUR FLOW
                    *       CONTIENT L'INFORMATION NECESSAIRE A COMPOSER
                    *       CETTE NOUVELLE DIRECTION.
                    *         (ON SUPPOSE QU'UN TOWARD STEP EST
                    *            TOUJOURS POSSIBLE)
                    *
                    *
                    *
                          COMMON/UNIT/CURNET,TABIN,LGFIL
                          COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)
                          COMMON/OD1/NCENT,GPQ(30,30),UPQ(30,30),KORIG(30)
                          COMMON /PARAM/ EPSI,TIME, MAXIT,PRINT,DEMAND,LIMKEEP
                          COMMON /INEXMS/IX(36),KEY
                          COMMON /GAMAS/GAMMA(35)
                    *
                          DIMENSION FLOW(400)
                          REAL MINSUM,MAXSUM,NEWSUM,SUMU1,SUMU2
                          LOGICAL AWAY

                          MAXSUM=0.0
                          MINSUM=1.0E99
                          DO 50  K=1,LIMKEEP
                          NEWSUM=0.0
                          CALL READMS(LGFIL,FLOW,400,K)

                          DO  10 I=1,NLINK
                       10 NEWSUM=NEWSUM+FLOW(I)*TRAV(I)
                          IF(MINSUM.LE.NEWSUM)GOTO 40
                          MINSUM=NEWSUM
                          KEYMIN=K

                       40 TETA=1.0/(1.0-GAMMA(K))
                          TETA1=TETA-1.0
                    *
                          Y=0.0
                          DO 110 II=1,NLINK
                          X=TETA1*(VCUM(II)-FLOW(II))
                          Y=Y+X*TRAV(II)
                      110 CONTINUE
                    *
                          IF(MAXSUM.LE.Y) GOTO 50
                    *
                          MAXSUM=Y
                          KEYMAX=K
                       50 CONTINUE
                    *
                    *   TRAITEMENT DE D(A)  EXP(T)    ( TOWARD STEP )
                    *
                          CALL READMS(LGFIL,FLOW,400,KEYMIN)
                          Y=0.0
                          DO  80  I=1,NLINK
                          X=FLOW(I)-VCUM(I)
```

```
      SUBROUTINE NEWFLOW    73/173  OPT=1                    FTN 4.6+460        21/06/7
                        FLOW(I)=X
                        Y=Y+X*TRAV(I)
   60             80 CONTINUE
                  *
                        AWAY=.FALSE.
                        NEWSUM=Y
                        KEYKEEP=KEYMIN
   65             *       TRAITEMENT DE D(A)  EXP(A)   ( AWAY STEP )
                  *
                  *
                  *    CRITERE DE CHOIX ENTRE  AWAY STEP ET TOWARD STEP
                  *     LORSQUE LES DEUX SONT POSSIBLES
   70             *
                        IF(MOD(NUMCYC,2).EQ.1) GOTO 150
                  *
                        IF(MAXSUM.GE.0.0) GOTO 150
                        CALL READMS(LGFIL,FLOW,400,KEYMAX)
   75                   TETA=1.0/(1.0-GAMMA(KEYMAX))
                        TETA1=TETA-1.0
                  *
                        DO 120 I=1,NLINK
                        FLOW(I)=TETA1*(VCUM(I)-FLOW(I))
   80             120 CONTINUE
                  *
                        NEWSUM=MAXSUM
                        AWAY=.TRUE.
                        KEYKEEP=KEYMAX
   85             *
                  150 CONTINUE
                        PRINT 9999,NEWSUM,MAXSUM,TETA
                 9999 FORMAT(/,1X,*  NEWSUM MAXSUM : *,E20.5,2X,E20.5,
                     +*  TETA : *,F10.6)
   90                   RETURN
                        END
```

```
SUBROUTINE SEARCH     73/173  OPT=1                    FTN 4.6+460        21/06/73

1                     SUBROUTINE SEARCH(STEP, DELTA)
                C
                      COMMON/NET1/ NNODE,NLINK,NOLAB(250),ND(400),TRAV(400),VCUM(400)
                      COMMON /LDATA/ TO(400),BETA(400),LTYP(400)
5                     DIMENSION DELTA(400), X(400)
                      COMMON/STAT/RUNSEC(10)
                      EQUIVALENCE (SHSEC,RUNSEC(3))
                C
                C
10                    CALL SECOND(TTIN)
                      PAS=1.00
                      S=0.0
                    5 PAS=PAS/10.
                      DO 10 I=1,10
15                    S=S+ PAS
                      DO 20 J=1,NLINK
                   20 X(J)=VCUM(J)+ S*DELTA(J)
                      CALL GRADY(VAL,X,DELTA,NLINK)
                      IF(VAL.GT.0.0)GOTO 30
20                  10 CONTINUE
                      STEP=1.0
                      GOTO 60
                   30 S = S - PAS
                      IF(PAS.LE.0.0001)50,5
25                 50 STEP= S + PAS*0.5
                      IF(STEP.GT.1.0)STEP=1.0
                C
                   60 CALL SECOND(TTOUT)
                      SHSEC=SHSEC+TTOUT-TTIN
30                    RETURN
                      END
```

ANNEXE D

Description des variables

DESCRIPTION DES VARIABLES

Remarque: Les descriptions des pages 30 et 74 ne figurent pas dans cette annexe.

Variable	Description
A	matrice de dimension $m \times n$.
A^i	la "ième" ligne de A, ("ième" contrainte).
b	vecteur de dimension $m \times 1$ (membre droit des contraintes).
b_i	la "ième" composante du vecteur b.
d	vecteur de dimension n représentant une direction de descente ($d \in E^n$).
d^*	une direction de descente particulière.
E^n	espace euclidien de dimension n.
$G(a)$	longueur de l'arc a.
$\overline{h}_{k,(pq)}$	une solution particulière permettant de composer une direction de descente par interpolation linéaire (voir $\overline{x}$)
$h^*_{k,(pq)}$	une solution particulière représentant un nombre de déplacements sur le chemin $k,(pq)$.
$h^*_{k^*,(pq)}$	une solution particulière représentant un nombre de déplacements sur le chemin particulier $k^*,(pq)$.
$h^{\ell}_{k,(pq)}$	une solution à l'itération ℓ représentant un nombre de déplacements sur le chemin $k,(pq)$.
$k^*,(pq)$	un chemin en particulier entre l'origine p et la destination q.
ℓ	compteur d'itérations.
L	nombre de points extrêmes (solutions antérieures) conservés pour effectuer la linéarisation interne du domaine de la fonction économique.

Variable	Description
P	ensemble permettant d'identifier les indices "j" tels que $\{x_j\}$ forme un ensemble de points de base pour la linéarisation interne du domaine de la fonction économique.
P^{ℓ}	contenu de l'ensemble P à l'itération ℓ.
t	variable réelle ($0 \leq t \leq 1$ pour la plupart des cas).
t^{ℓ}	valeur de t à l'itération ℓ.
v_a^{ℓ}	quantité totale de déplacements sur l'arc a pour la solution à l'itération ℓ.
$\hat{v}_a$	une solution particulière réalisable.
$\bar{v}_a$	solution obtenue lors de la recherche d'une direction de descente par approximation linéaire (résolution du problème III.2.3).
$\bar{v}_a^j$	une solution particulière (j "ième"), réalisable.
x	un vecteur de dimension n x 1 ($x \in E^n$).
x_j	la j"ième" composante de x.
$\bar{x}$	solution du sous-problème: $\underset{\{x \in \underline{X}\}}{\text{Min}} \ \nabla f(x^{\ell})\ (x - x^{\ell})$ le vecteur $d = (\bar{x} - x^{\ell})$ forme une direction de descente lorsque $\nabla f(x^{\ell})\ (\bar{x} - x^{\ell}) < 0$.
$x^{\ell}, \hat{x}^{\ell}$	vecteur particulier réalisable ($x^{\ell}, \hat{x}^{\ell} \in \underline{X}$) à l'itération ℓ.
$\bar{x}^{\ell}$	valeur de $\bar{x}$ à l'itération ℓ.
x^j	vecteur particulier réalisable ($x^j \in \underline{X}$)
X	vecteur de dimension L dont chaque composante est un vecteur x^j $j=0,\ldots,L-1$ de dimension n. $X = (x^0, x^1, \ldots, x^{L-1})$

Variable	Description
$\overline{X}$	domaine réalisable pour la fonction économique.
α	vecteur $(\alpha_0,\alpha_1,\ldots,\alpha_{L-1})$ contenant les coefficients associés aux L points de base de la linéarisation interne du domaine de la fonction économique. On a la relation: $\sum_{j=0}^{L-1} \alpha_j = 1$.
αX	produit vectoriel entre α et X : $\alpha X = \sum_{j=0}^{L-1} \alpha_j x^j$.
$\hat{\alpha}$	vecteur α particulier.
$\hat{\alpha},\hat{\alpha}^{\ell}$	vecteurs α particuliers.
$\delta_{ak,(pq)}$	fonction indicatrice associée à la solution courante $\hat{v}_a$.
$\nabla f(x)$	gradient de la fonction f(x): $\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \ldots, \frac{\partial f(x)}{\partial x_n}\right)$

REMERCIEMENTS

Avant tout, je tiens à remercier Monsieur Michael Florian, directeur de ce stage, pour son importante contribution. Sur le plan professionnel, ses apports théoriques, sa bibliographie et sa compréhension ont rendu possible ce travail; il a pris à coeur ce travail et je luis suis reconnaissant pour le temps qu'il y a consacré. Sur le plan personnel, j'apprécie l'attitude compréhensive et la patience qu'il a manifestées à mon égard ainsi que la liberté qu'il m'a laissée.

J'en profite pour remercier tous mes compagnes et compagnons, amies et amis de classe et de travail pour le soutien social et parfois académique qu'ils m'ont apporté.

Enfin, je souligne l'excellent travail de dactylographie de Madame Madeleine Constantin.

BIBLIOGRAPHIE

[1] Cantor, D.G., Gerla, M., "Optimal Routing in a Packet-Switched Computer-Network", IEEE Transaction on Computers C-23, Vol. 10, pp. 1062-1068, 1974.

[2] Dantzig, G.B., Linear Programming and Extentions, Princeton University Press, 1963.

[3] Dial, R., Glover, F., Karney, D., Klingman, D., "A Computational Analysis of Alternative Algorithms and Labelling techniques for finding shortest paths trees". Research Report CCS 291, 1977.

[4] Dreyfus, E. Stuard, "An Appraisal of Some Shortest-Path Algorithms" A.M. Geoffrion - Perspectives on optimisation, Addison-Wesley, 1972.

[5] Ferland, J.A., "Eléments de programmation nonlinéaire". Document de travail # 38, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1973.

[6] Florian, M., "An Improved Linear Approximation Algorithm for the Network Equilibrium (packet-switching) Problem", IEEE Control Systems Society, 1977 (FA 1-0:00)

[7] Frank, M. Wolfe, P., "An Algorithm for Quadratic Programming", Naval Research Logistics Quarterly, Vol. 3, pp. 95-110, 1956.

[8] Geoffrion, A.M."Elements of Large-Scale Mathematical Programming", Management Science, 16 (11), 1970.

[9] Geoffrion, A.M. "Duality in Nonlinear Programming: A Simplified Applications-Oriented Development",SIAM Review 13 (1), 1971.

[10] Goldman, A.J., "Resolution and Separation Theorems for Polyhedral Convex Sets", H.W. Kuhn and A.M. Tucker (eds.), Linear Inequalities and Related Systems . Princeton University Press, Princeton, New Jersey, 1956.

[11] Hearn, D., Kuhn, H., Lasaga, F., "Pathfix: A Fixed Point Method for Computing Traffic Equilibria", Mathtec Inc., Technical Report, 1977.

[12] Holloway, C.A., "An Extension of the Frank and Wolfe Method for Feasible Directions", Mathematical Programming, Vol. 6, pp. 14-27, 1974.

[13] Hu, T.C., Integer Programming and Network Flows. Addison-Wesley, 1970.

[14] Lasdon, L.S., Optimisation Theory for Large Systems. Collier-MacMillan Limited, London, 1970.

[15] LeBlanc, L.J., "Mathematical Programming Algorithms for Large Scale Network Equilibrium and Network Design Problems". Ph.D. dissertation, Department of Industrial Engineering and Management Sciences, Northwestern University, 1973.

[16] Luenberger, D.G., Introduction to Linear and Nonlinear Programming. Addison-Wesley, Reading, Mass, 1973.

[17] Nguyen, S., "A Mathematical Programming Approach to Equilibrium of Traffic Assignment with Fixed Demands", Publication # 18, Centre de recherche sur les transports, Université de Montréal, 1973.

[18] Nguyen, S. "Une approche unifiée des méthodes d'équilibre pour l'affectation du trafic". Centre de recherche sur les transports, Université de Montréal, mars 1974.

[19] Nguyen, S., James, L., "Traffic - An Equilibrium Traffic Assignment Program", Publication # 17, Centre de recherche sur les transports, Université de Montréal, 1975.

[20] Pape, U., "Implementation and Efficiency of Moore-Algorithms for the Shortest Route Problem", Mathematical Programming, 7, pp. 212-222, 1974.

[21] Potts, O., Oliver, R., Flows in Transportation Network. Academic Press, 1972.

[22] Steenbrink, P.A., Optimisation of Transportation Networks. John Wiley, 1974.

[23] Wolfe, P., Convergeance Theory in Nonlinear Programming, in: Integer and Nonlinear Programming" Ed. J. Abadie, North-Holland, Amsterdam, 1970.

[24] Zangwill, W., Nonlinear Programming: A Unified Approach. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1969.

[25] Zoutendijk, Methods of Feasible Directions, a Study in Linear and Nonlinear Programming. Elsevier, Amsterdam, 1960.